IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Richard J. Ollio
4/21/2022

Outline

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

Executive Summary

- Summary of methodologies
  - The project focused on three main methodologies in order to complete the scope of the project. They are: wrangling and cleaning the data (Exploratory Data Analysis), visualizing the data effectively, and using predictive analytics to attempt to predict if a launch was going to fail, and to try and figure out the main factors that are contributing to the success or failure of a given mission.
- Summary of all results
  - By completing the above methodologies we found that there was a definitive correlation between some of the factors and predicting mission failure/success. This is shown by visualizing the data effectively and respective predictive analytics in trying to predict mission failure.

Introduction

- Project background and context
  - The project is looking at SpaceX Reusable rocket launches and trying to determine why a mission is met with success or failure. Falcon 9 rocket launches cost approximately $62 million, so success or failure is a very costly factor. SpaceY wants to use this data to effectively compete with SpaceX and break into the commercial rocket launching business. We will do this by reworking and exploring the SpaceX data, visualizing it effectively, and then to use the Space X data to predict what makes a rocket mission successful or a failure, so we can better engineer our rockets to have the most success possible.
- Problems you want to find answers
  - What causes a rocket launch to land successfully or not?
  - What are the biggest factors when launching and landing so we can reuse the first stage?
  - What is the price of each launch by each launch type?

Section 1

Methodology

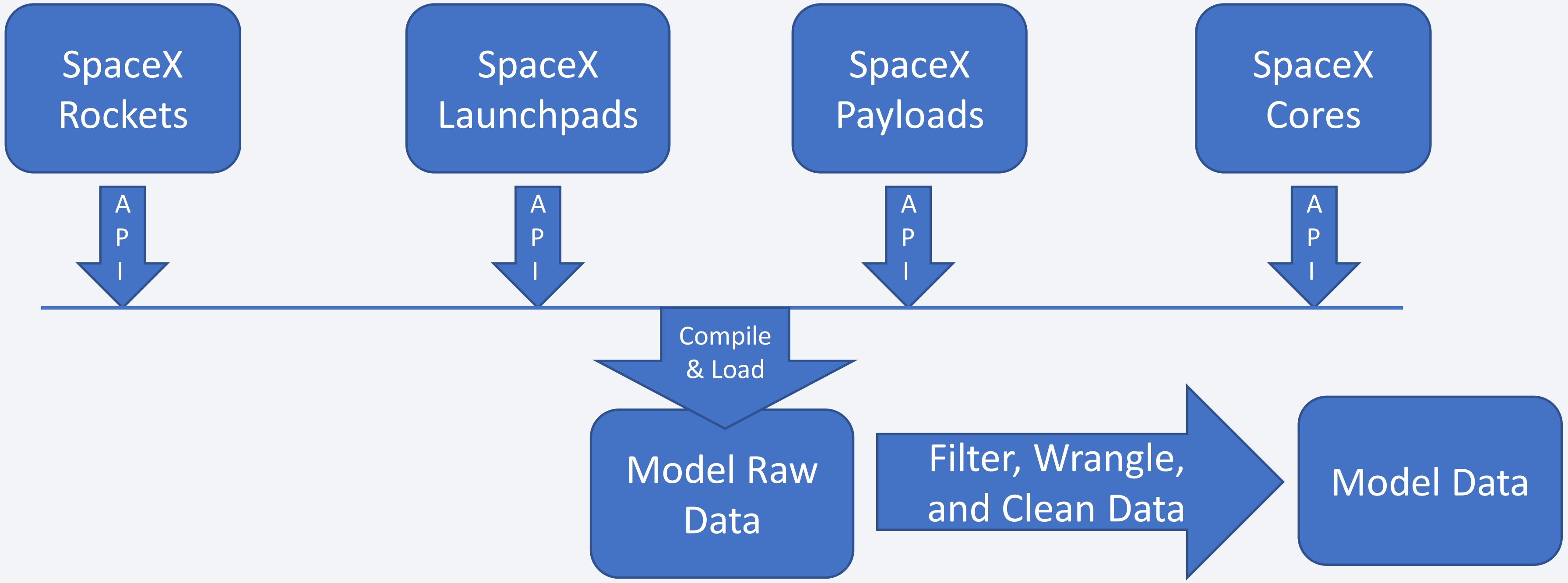
Methodology

Executive Summary

- Data collection methodology:
  - Data was collection by webscaping spacexdata.com using an API call.
- Perform data wrangling
  - Data was standardized by dealing with missing values in the data set and by making sure that values were in the correct data type.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - By using different methods of classification and by measuring their individual effectivness we can find the best classification model to use for our data model.

Data Collection

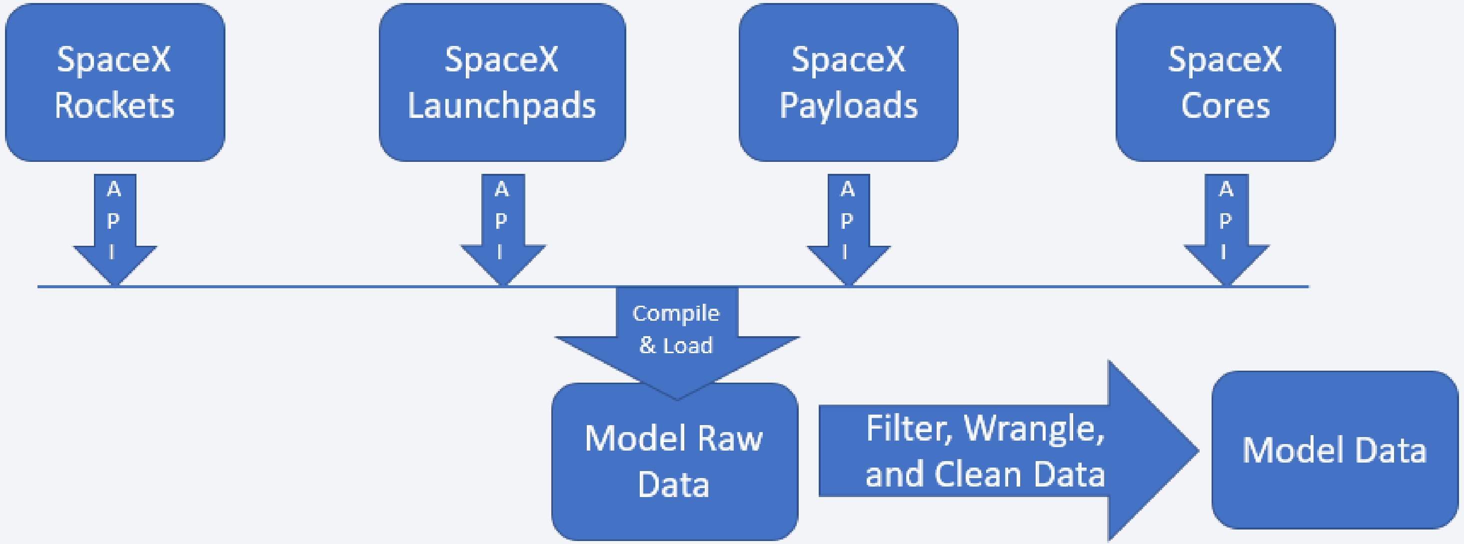
- Data sets were collected from the spacexdata.com site using an API call to the site to web scrape the site and load in the data. This was split into grabbing the rockets, launchpads, payloads, and cores.
- You need to present your data collection process use key phrases and flowcharts



Data Collection – SpaceX API

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

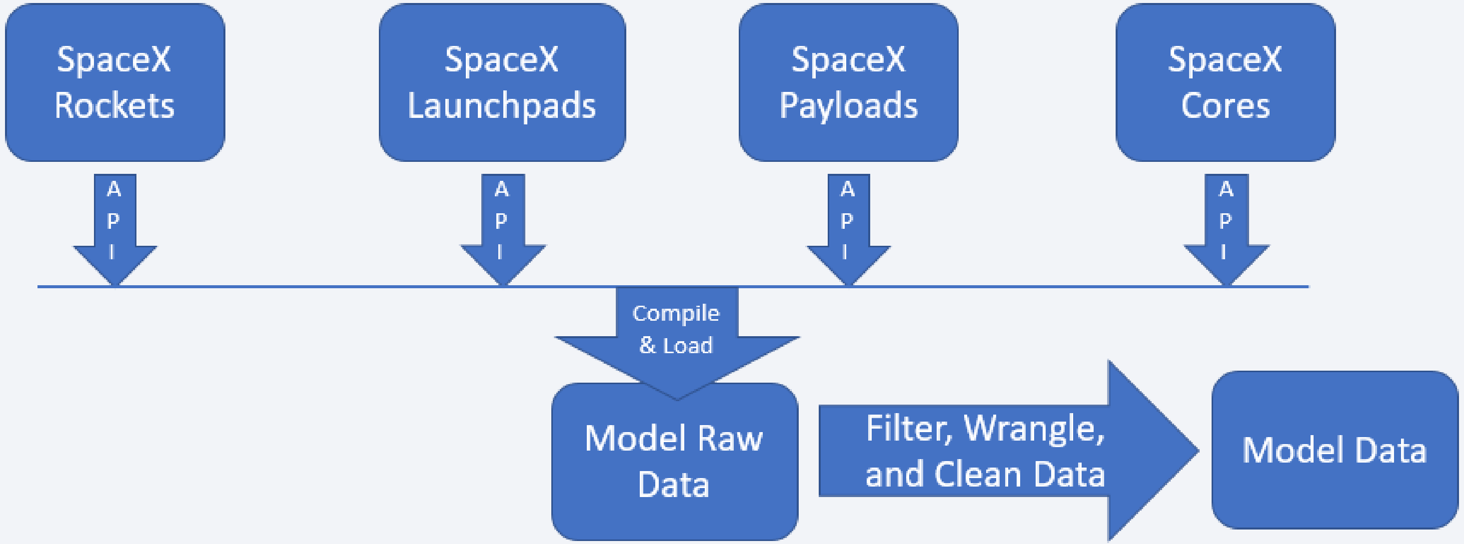
- Add the GitHub URL of the completed SpaceX API calls notebook (must include comple code cell and outcome cell), as external reference and peer-re purpose



Data Collection - Scraping

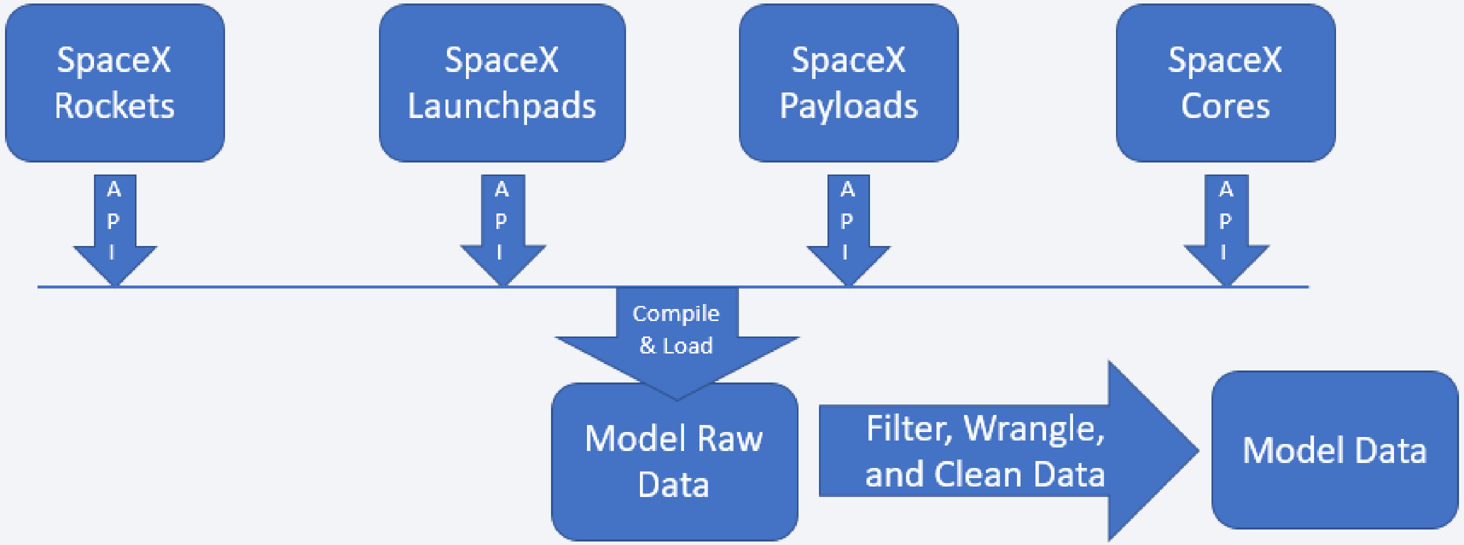
- Present your web scraping process using key phrases and flowcharts

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



Data Wrangling

- Data collection methodology:
    - Data was collection by webscaping spacexdata.com using an API call.
- Perform data wrangling
    - Data was standardized by dealing with missing values in the data set and by making sure that values were in the correct data type.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - By using different methods of classification and by measuring their individual effectivness we can find the best classification model to use for our data model.



EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
  - Flightnumber by Payload Mass Scatterplot – To show that across flight numbers/time how the payload mass changed over time and how that affected the failure or success numbers – mass increases over time and failure rate decreases over time.
  - Flightnumber by Launch Site Scatterplot – To show how flight number and the site affected the success/failure – VAFB SLC stopped being used and KSC LC 38A was started after the others. The large % of failures occurred on the CCAPS SLC 40 site.
  - Payload vs Launch Site Scatter – Payloads under 6000 kg held most of the failures.
  - Ploted bar chart of Orbit and Outcome% - GTO, ISS are worst performers at <60%.

- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - Displayed Unique Launch Sites
  - Displayed a limited set of records (5) where the launch site begins with ‘CCA’
  - Displayed the total payload mass of NASA launched boosters
  - Displayed average payload mass carried by booster F9 v1.1
  - Display the date of the first successful ground pad landing
  - Display names of boosters with success in drone ship and payload mass between 4000 and 6000 kg
  - List total number of successful and failure mission outcomes
  - List booster versions which carried the max payload mass
  - List failed landing outcomes in drone ship, booster versions, and launch sites for year 2015
  - Rank count of landing outcomes between June 4, 2010 and March 20, 2017 desc.

- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

Build an Interactive Map with Folium

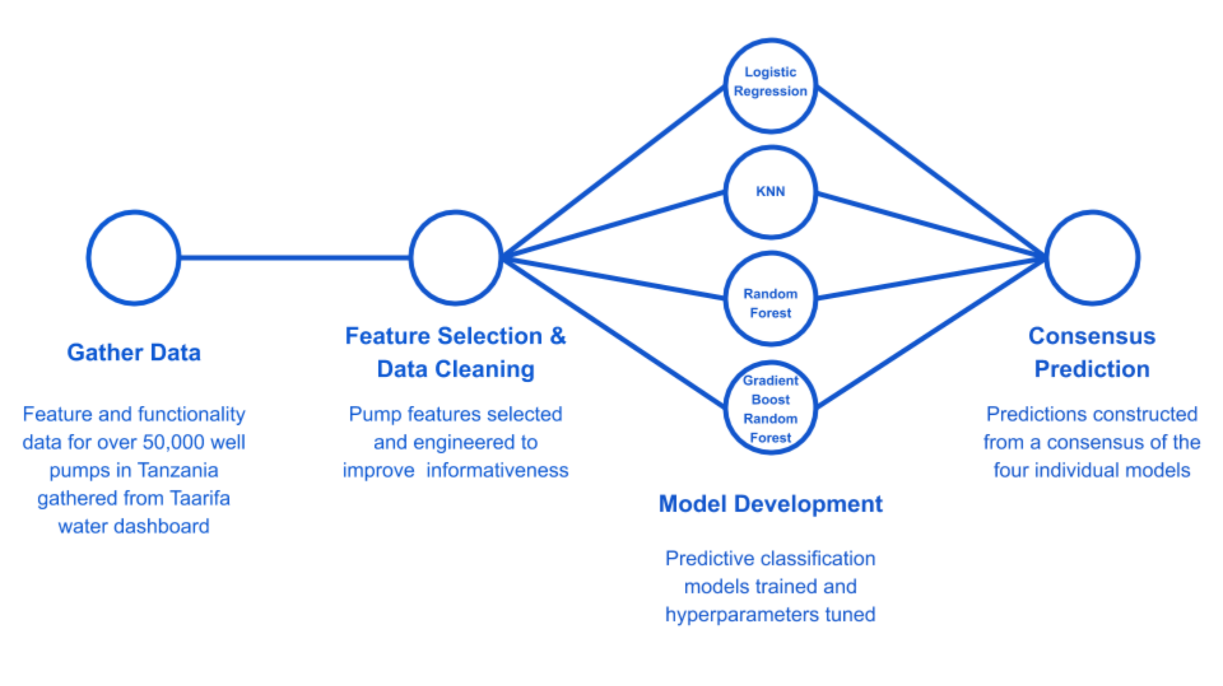
- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map and explain why
    - Added markers and circles to show different launch sites on the map and eventually grouping them together as clusters with number and color showing how many launches were from each launch site and the average success of the launch site by color (green, yellow, red to show good to bad results on average) This was started by giving green and red distinctions to single launches to show pass/fail. Also lines were added to show distance between the launch site and certain objects such as railroads, ocean, highway, city-center, etc.
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with Plotly Dash

- I created the US Domestic Airline Flights Performance Dashboard where you can filter by report (dashboard) and the year. Some of the visualizations created were average carrier delay time by airline (line), average weather delay time, Average NAS delay time, flight count by airline to dest. State (heatmap box), % of flights by reporting airline (pie), and number of flights from origin state (geo map)
- These were created to give a holistic overview of the airline data we had available over each different year.

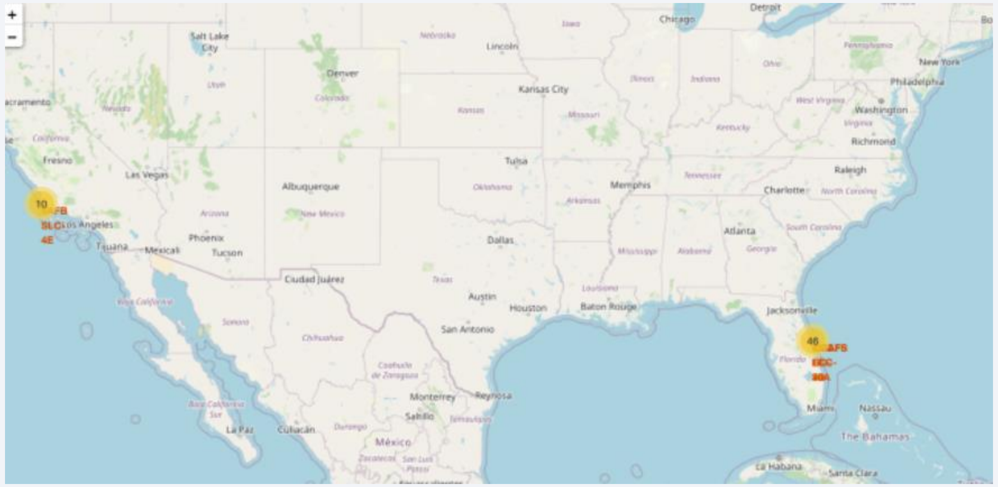
Predictive Analysis (Classification)

- We tested each different type of classification model and then calculated the accuracy of each model and what the R-squared value was for each value, the best classification model and the one that was chosen was the one with the lowest R-squared and highest accuracy.



Results

- Through exploratory analysis we found what are the most likely factors that cause the failure or success of a rocket phase one landing. This also helped with understanding the data as well as quantifying and qualitatively understanding it. Also, it gave an understanding of what the makeup of the data was, data types, etc. so we would know how to proceed with the data for analysis.
- We saw the outcome of the predictive analysis to be that the best model was the decision tree at 83.33%.

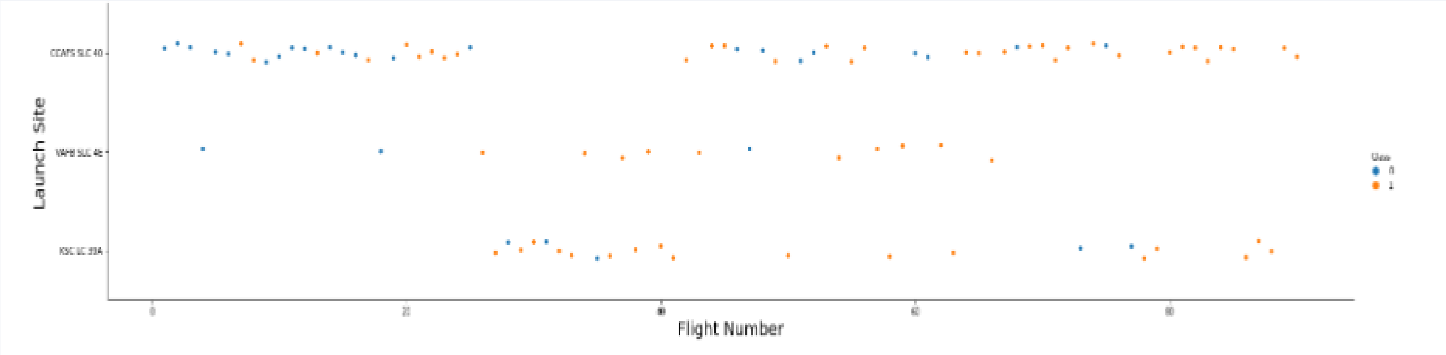


Section 2

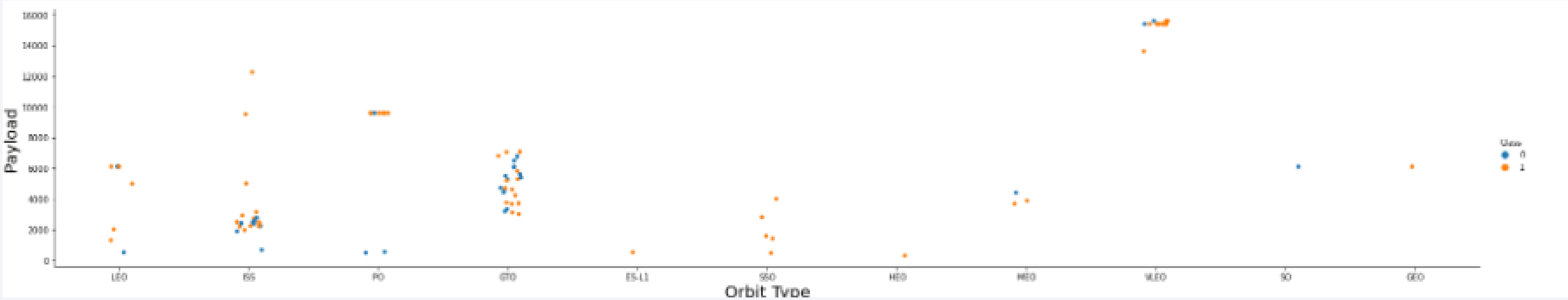
Insights drawn from EDA

Flight Number vs. Launch Site

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

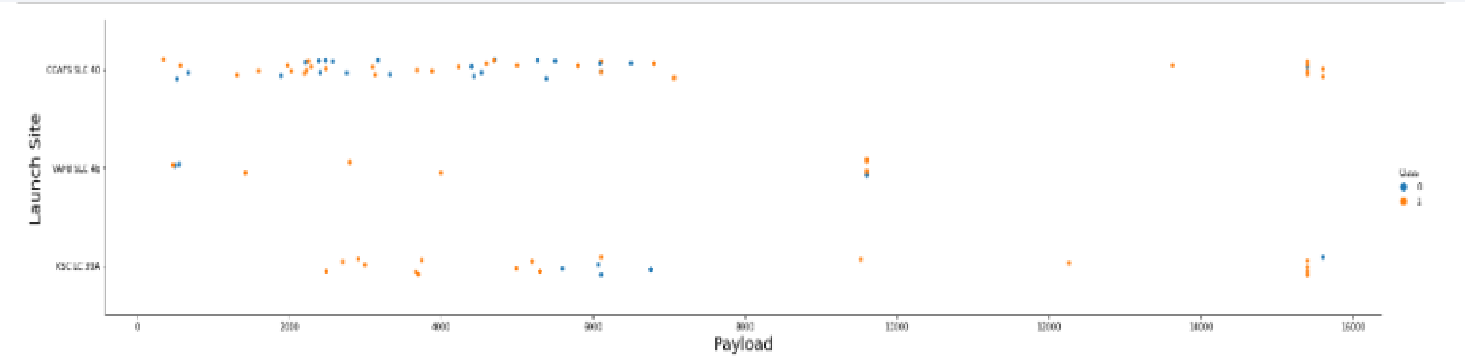


- Show the screenshot of the scatter plot with explanations

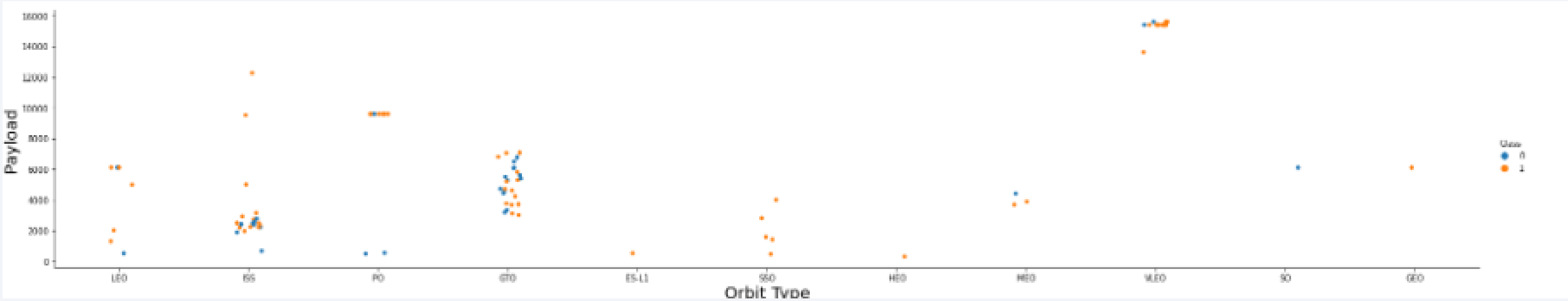


Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

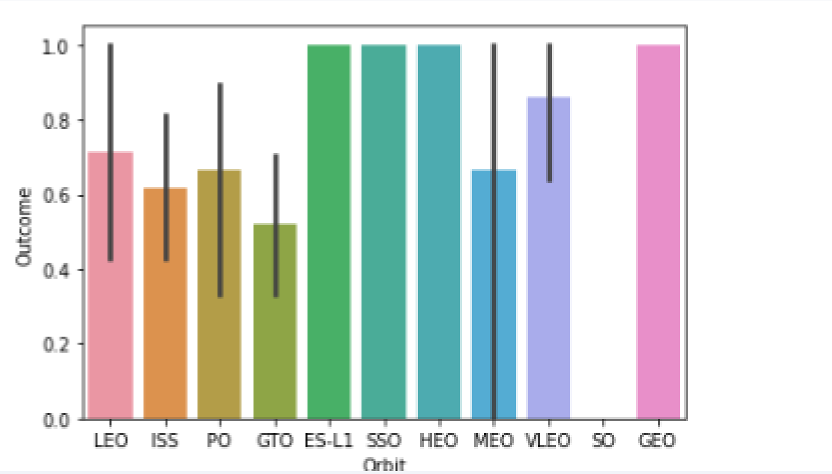


- Show the screenshot of the scatter plot with explanations

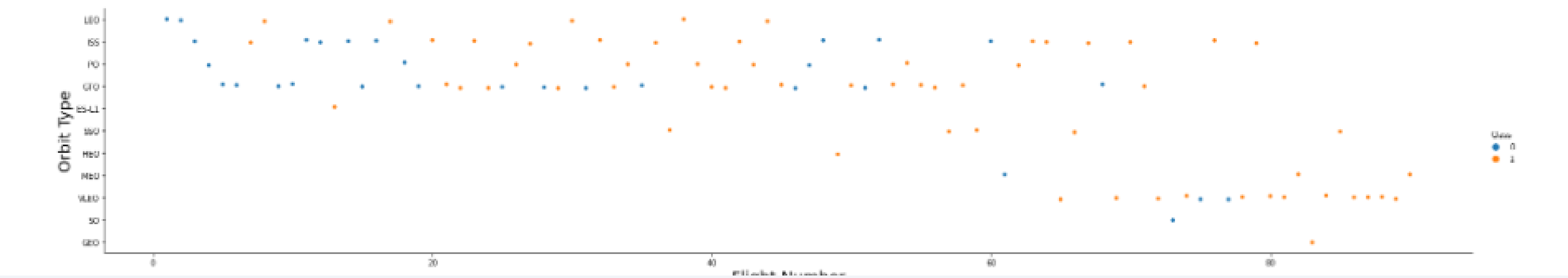


Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

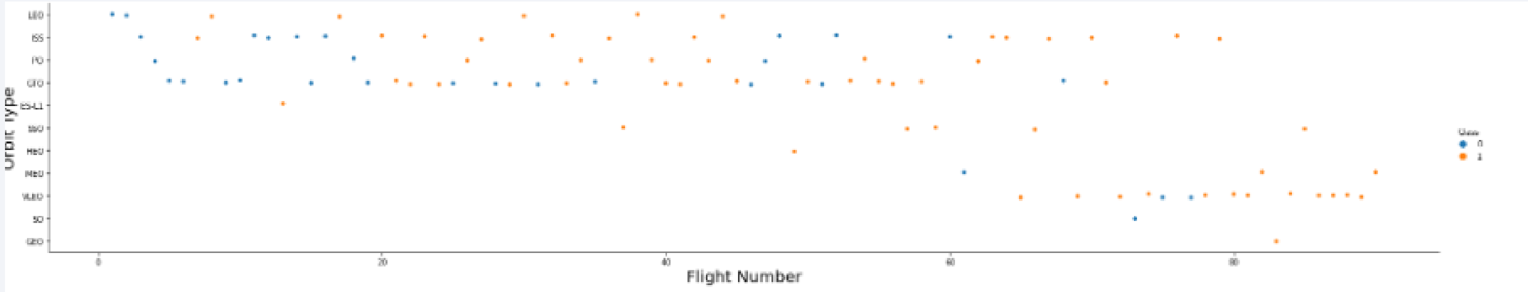


- Show the screenshot of the scatter plot with explanations

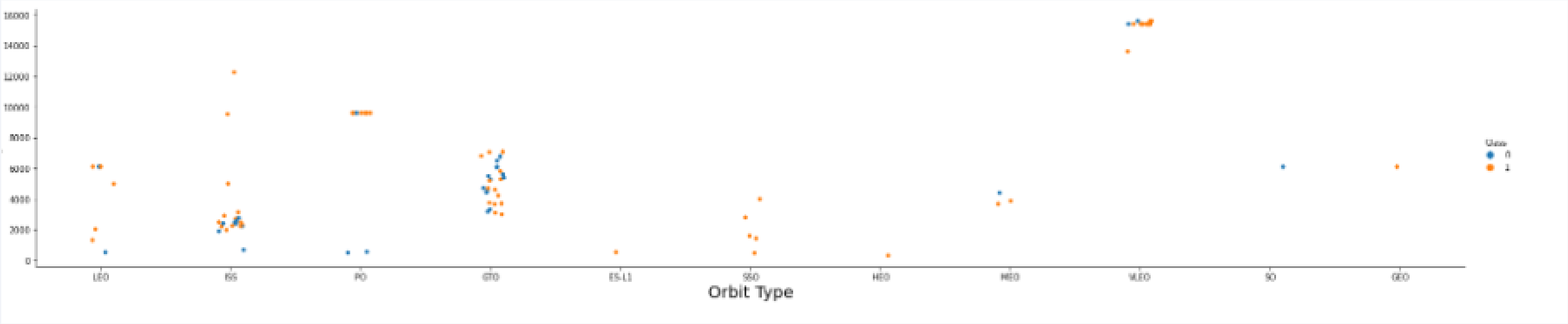


Flight Number vs. Orbit Type

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



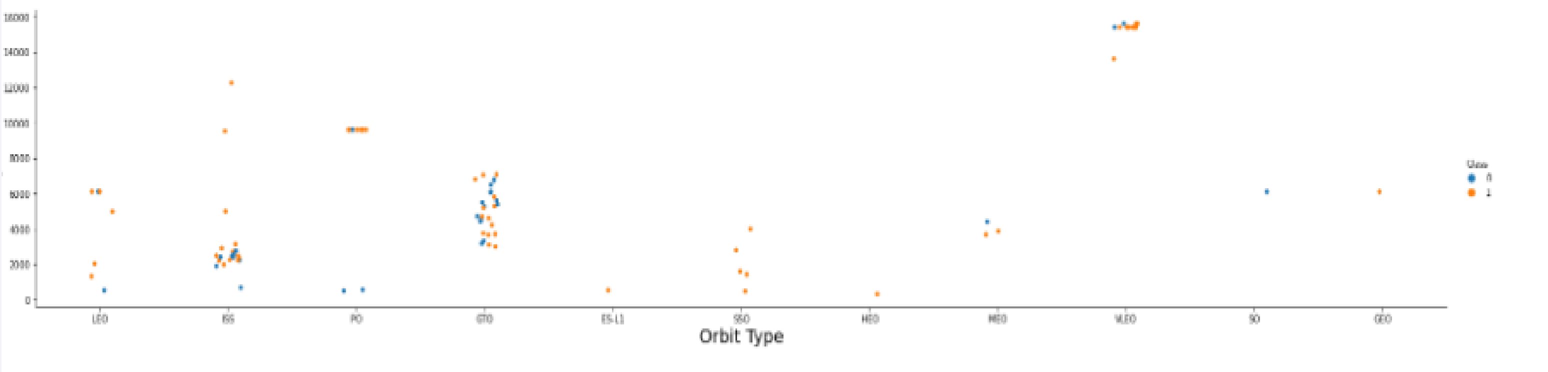
- Show the screenshot of the scatter plot with explanations

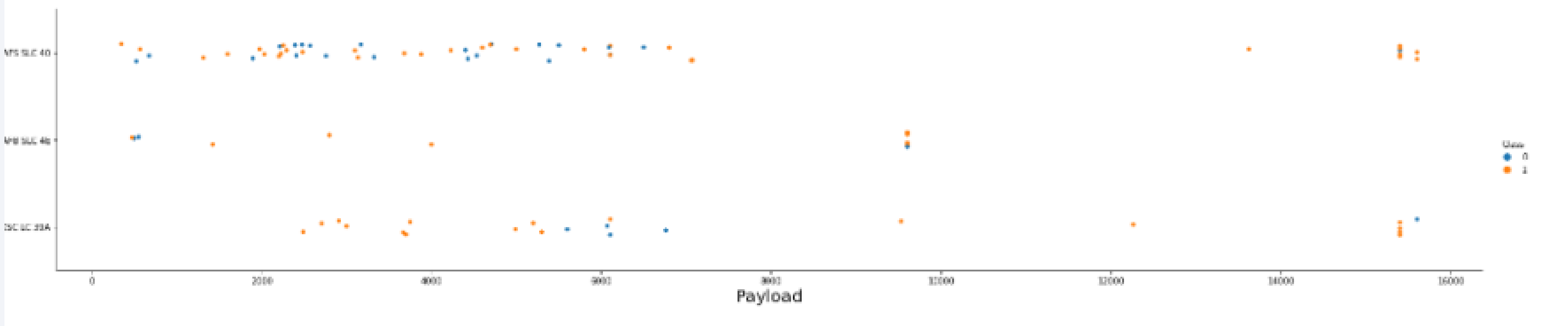


Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

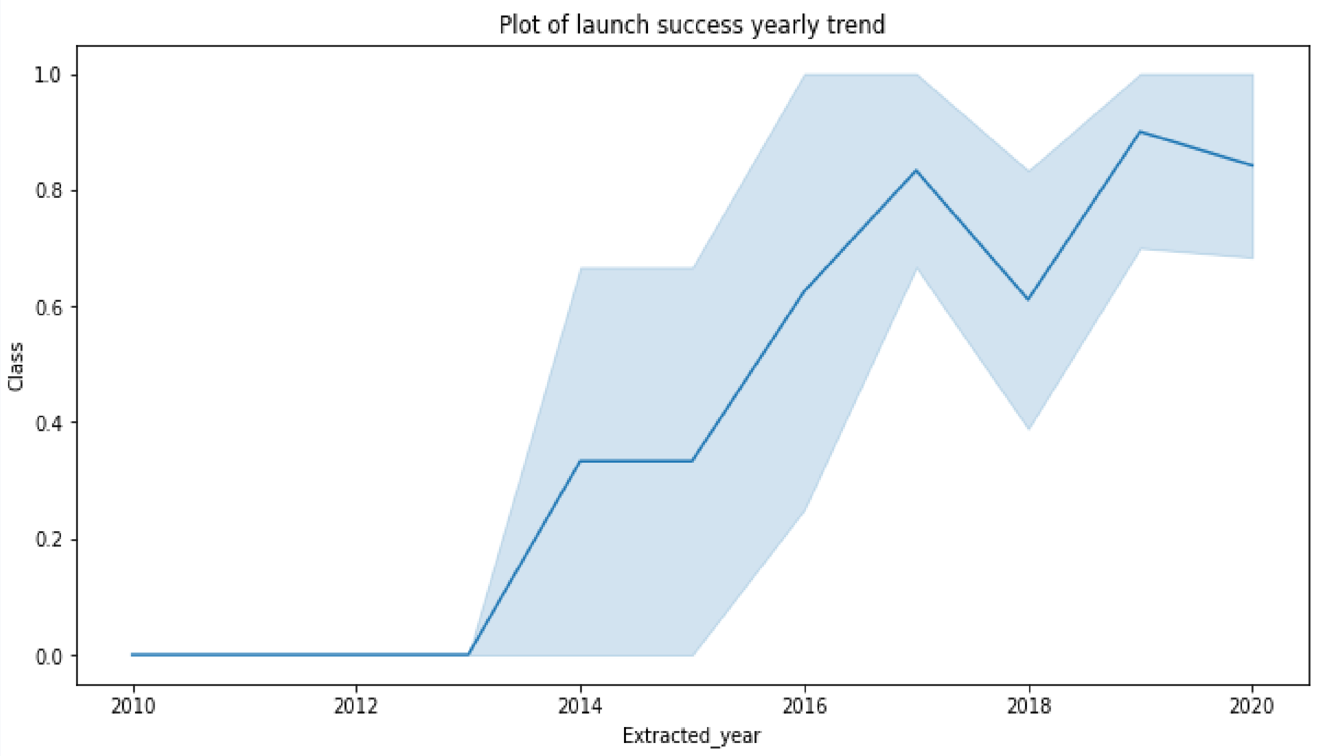
- Show the screenshot of the scatter plot with explanations





Launch Success Yearly Trend

- From 2010 to 2013 there was no increase/movement in success. Starting in 2013 launch success increased until 2017, when it took a small tumble back down to around 50% on its way up to 2020 where the success rate is 80%.



All Launch Site Names

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

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	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

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

In [11]:

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	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

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

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

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

In [13]:

```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

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

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

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

List the total number of successful and failure mission outcomes

In [16]:

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

Out[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- We used the maximum function (MAX()) to tell what the maximum payload was.

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

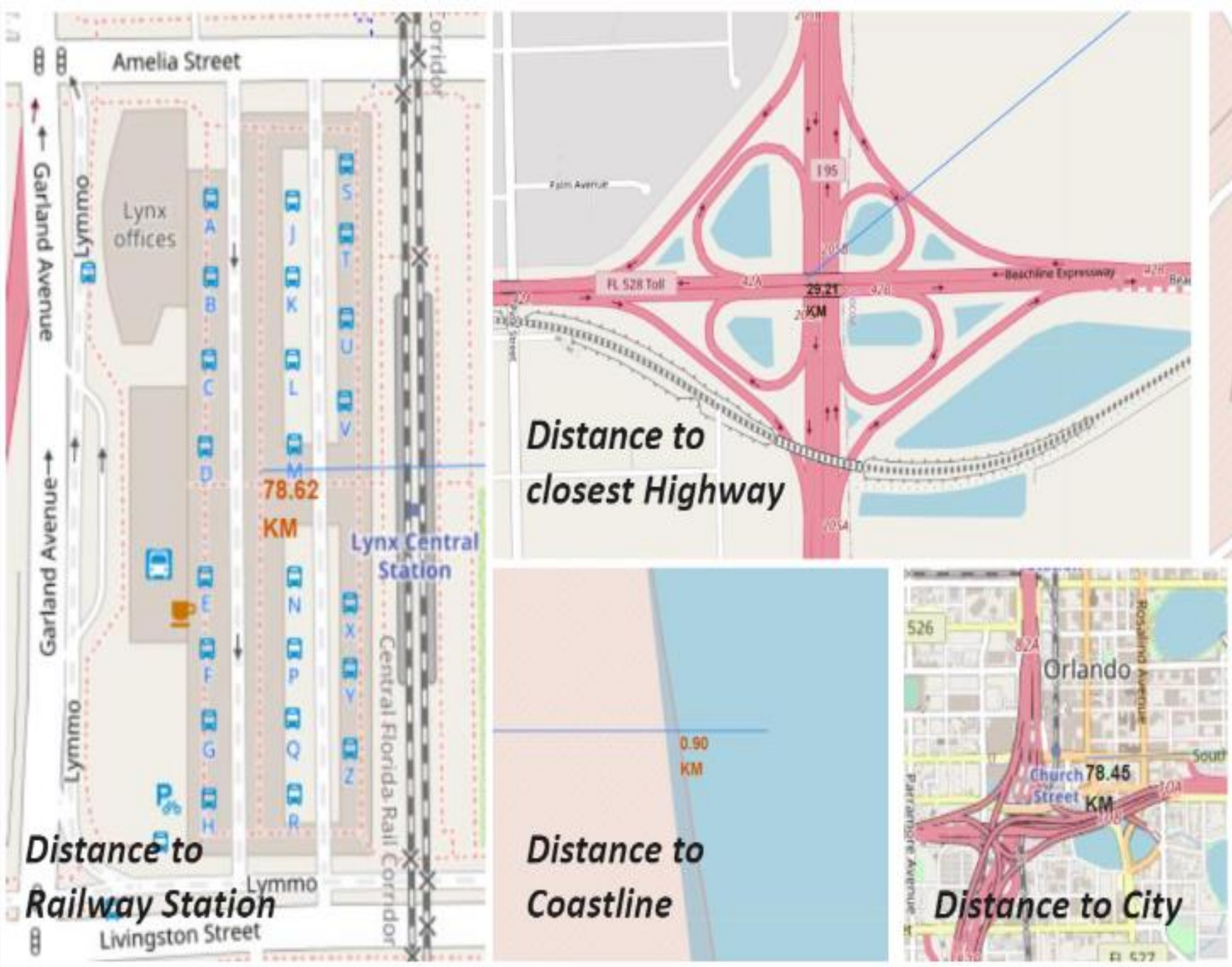
<Folium Map Screenshot 2>

Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

<Folium Map Screenshot 3>

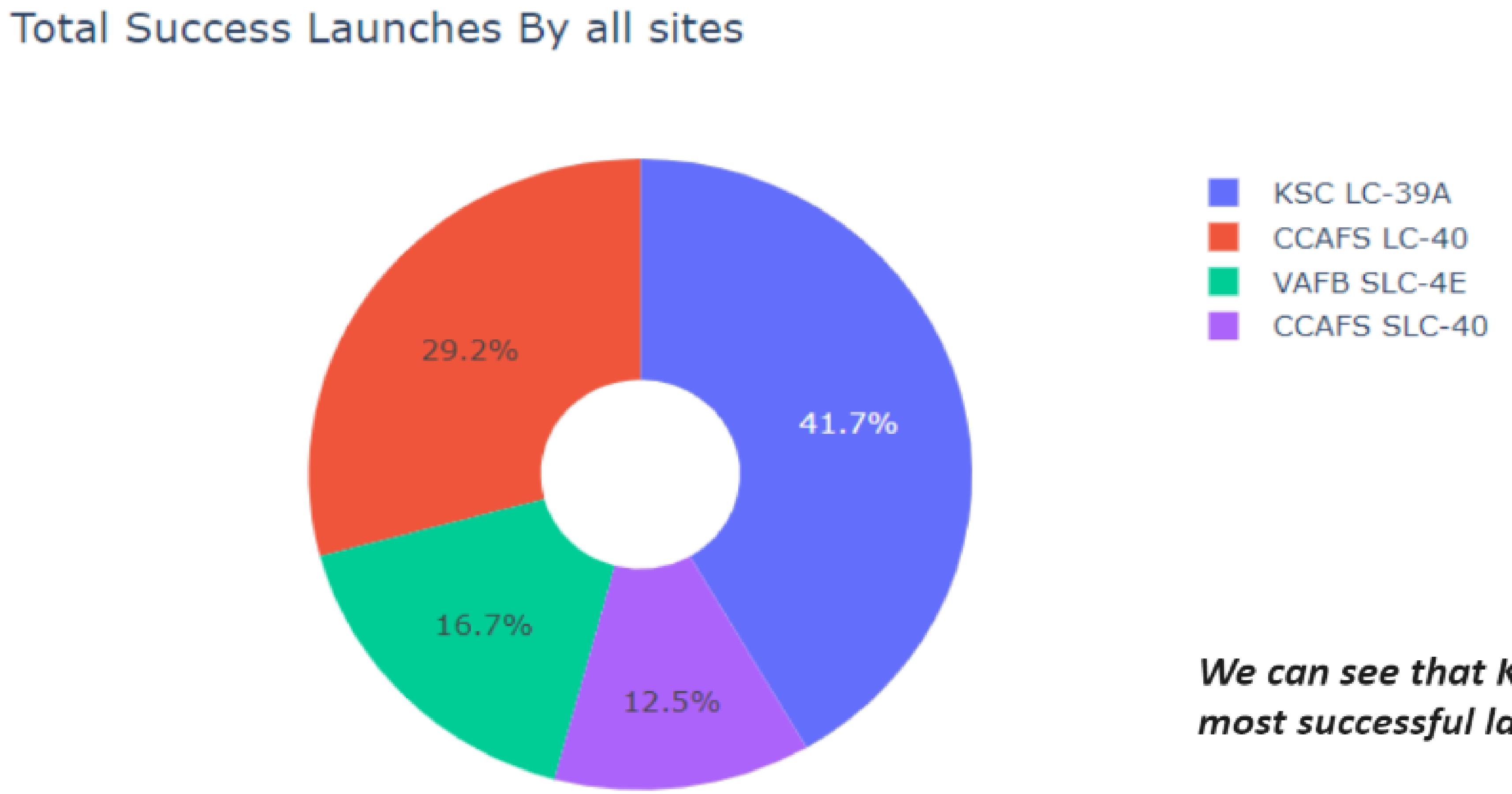


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

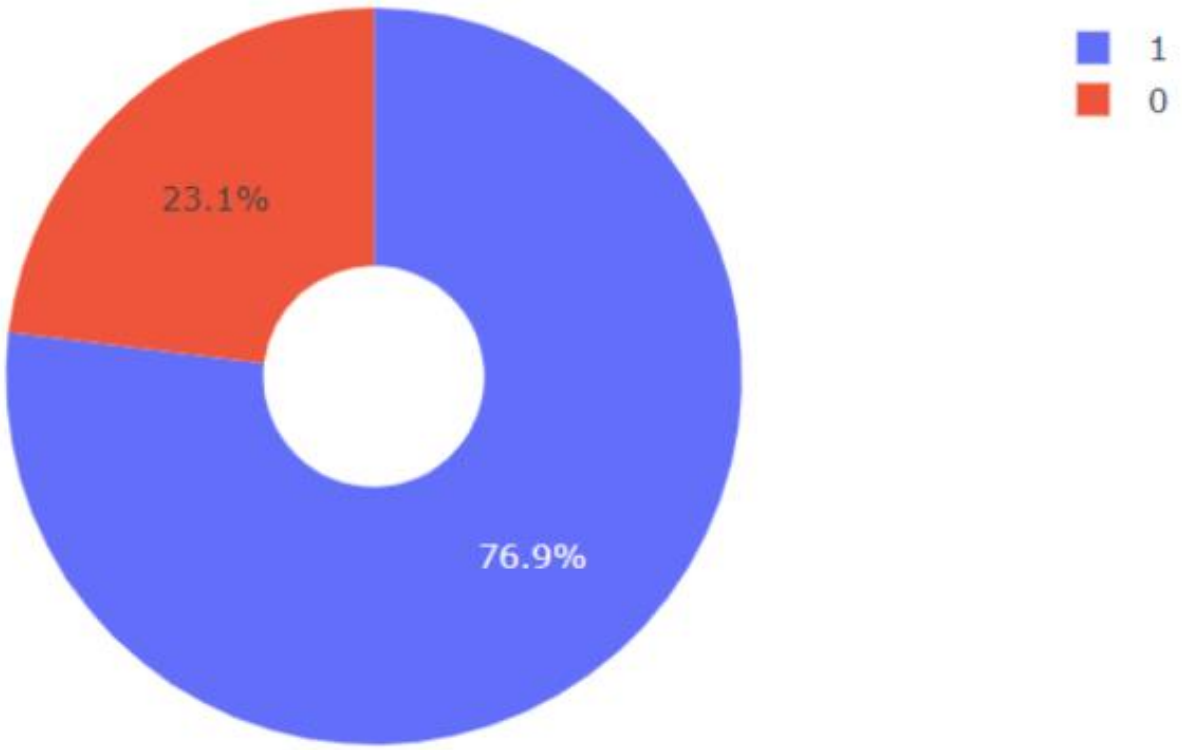
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



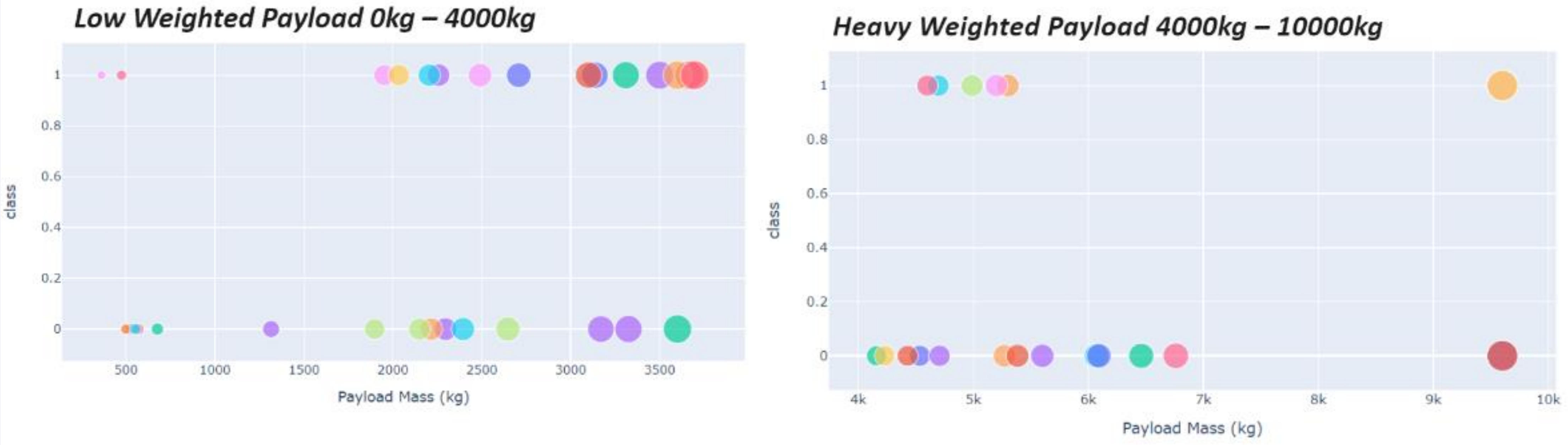
We can see that KSC LC-39A had the most successful launches from all the sites

<Dashboard Screenshot 2>



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

<Dashboard Screenshot 3>



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

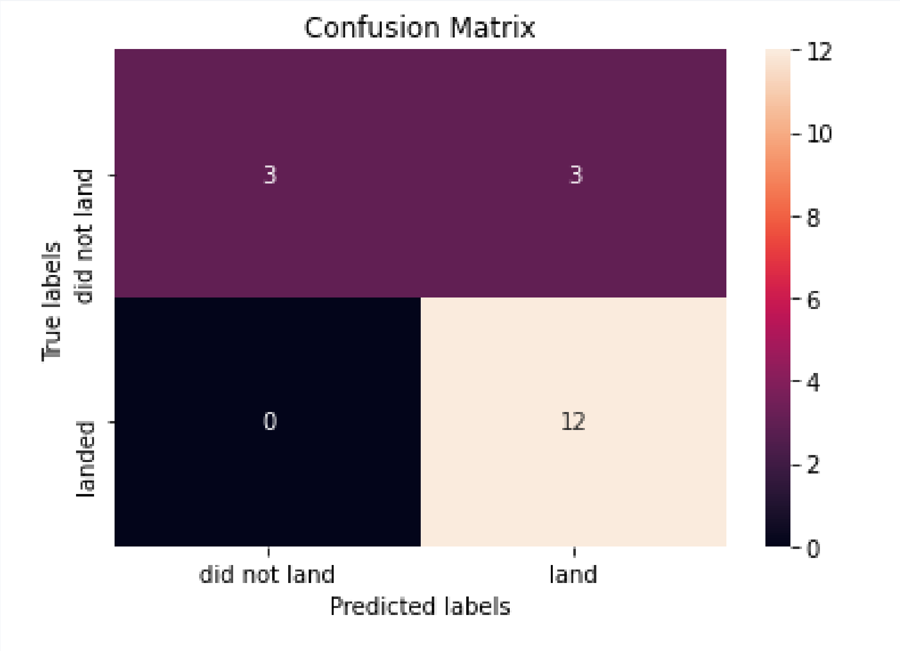
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation



Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

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

```
import dash
from dash import html
from dash import dcc
from dash.dependencies import Input, Output, State
import plotly.graph_objects as go
import plotly.express as px
from dash import no_update

# Create a dash application
app = dash.Dash(__name__)

# REVIEW1: Clear the layout and do not display exception till callback gets executed
app.config.suppress_callback_exceptions = True

# Read the airline data into pandas dataframe
airline_data =  pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/airline_data.csv',
                            encoding = "ISO-8859-1",
                            dtype={'Div1Airport': str, 'Div1TailNum': str,
                                   'Div2Airport': str, 'Div2TailNum': str})


# List of years
year_list = [i for i in range(2005, 2021, 1)]

"""Compute graph data for creating yearly airline performance report

Function that takes airline data as input and create 5 dataframes based on the grouping condition to be used for plottling charts and grphs.

Argument:

    df: Filtered dataframe

Returns:
   Dataframes to create graph.
"""
def compute_data_choice_1(df):
    # Cancellation Category Count
    bar_data = df.groupby(['Month','CancellationCode'])['Flights'].sum().reset_index()
    # Average flight time by reporting airline
    line_data = df.groupby(['Month','Reporting_Airline'])['AirTime'].mean().reset_index()
    # Diverted Airport Landings
    div_data = df[df['DivAirportLandings'] != 0.0]
    # Source state count
    map_data = df.groupby(['OriginState'])['Flights'].sum().reset_index()
    # Destination state count
    tree_data = df.groupby(['DestState', 'Reporting_Airline'])['Flights'].sum().reset_index()
    return bar_data, line_data, div_data, map_data, tree_data


"""Compute graph data for creating yearly airline delay report

This function takes in airline data and selected year as an input and performs computation for creating charts and plots.

Arguments:
    df: Input airline data.

Returns:
    Computed average dataframes for carrier delay, weather delay, NAS delay, security delay, and late aircraft delay.
"""
def compute_data_choice_2(df):
    # Compute delay averages
    avg_car = df.groupby(['Month','Reporting_Airline'])['CarrierDelay'].mean().reset_index()
    avg_weather = df.groupby(['Month','Reporting_Airline'])['WeatherDelay'].mean().reset_index()
    avg_NAS = df.groupby(['Month','Reporting_Airline'])['NASDelay'].mean().reset_index()
    avg_sec = df.groupby(['Month','Reporting_Airline'])['SecurityDelay'].mean().reset_index()
    avg_late = df.groupby(['Month','Reporting_Airline'])['LateAircraftDelay'].mean().reset_index()
    return avg_car, avg_weather, avg_NAS, avg_sec, avg_late


# Application layout
app.layout = html.Div(children=[
                                # TASK1: Add title to the dashboard
                                # Enter your code below. Make sure you have correct formatting.
                                html.H1('US Domestic Airline Flights Performance',
                                style={'textAlign': 'center', 'color': '#503D36', 'font-size': 24}),
                                # REVIEW2: Dropdown creation
                                # Create an outer division
                                html.Div([
                                    # Add an division
                                    html.Div([
                                        # Create an division for adding dropdown helper text for report type
                                        html.Div(
                                            [
                                            html.H2('Report Type:', style={'margin-right': '2em'}),
                                            ]
                                        ),
                                        # TASK2: Add a dropdown
                                        # Enter your code below. Make sure you have correct formatting.
                                                      dcc.Dropdown(id='input-type',
                  options=[
                          {'label': 'Yearly Airline Performance Report', 'value': 'OPT1'},
                          {'label': 'Yearly Airline Delay Report', 'value': 'OPT2'}
                          ],
                 placeholder='Select a report type',
                 style={'minWidth': '80%', 'padding': '3px', 'font-size': 20, 'textAlign': 'center'})
                                    # Place them next to each other using the division style
                                    ], style={'display':'flex'}),

                                    # Add next division
                                    html.Div([
                                       # Create an division for adding dropdown helper text for choosing year
                                        html.Div(
                                            [
                                            html.H2('Choose Year:', style={'margin-right': '2em'})
                                            ]
                                        ),
                                        dcc.Dropdown(id='input-year',
                                                     # Update dropdown values using list comphrehension
                                                     options=[{'label': i, 'value': i} for i in year_list],
                                                     placeholder="Select a year",
                                                     style={'width':'80%', 'padding':'3px', 'font-size': '20px', 'text-align-last' : 'center'}),
                                            # Place them next to each other using the division style
                                            ], style={'display': 'flex'}),
                                          ]),

                                # Add Computed graphs
                                # REVIEW3: Observe how we add an empty division and providing an id that will be updated during callback
                                html.Div([ ], id='plot1'),

                                html.Div([
                                        html.Div([ ], id='plot2'),
                                        html.Div([ ], id='plot3')
                                ], style={'display': 'flex'}),

                                # TASK3: Add a division with two empty divisions inside. See above disvision for example.
                                # Enter your code below. Make sure you have correct formatting.
                                 html.Div([
                                        html.Div([ ], id='plot4'),
                                        html.Div([ ], id='plot5')
                                ], style={'display': 'flex'}),
                                ])

# Callback function definition
```

Thank you!