IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Nor-Arevian Georgii>
<2/18/2025>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.
- We will predict if the Falcon 9 first stage will land successfully. What launch features impact for success landing.

Section 1

Methodology

Data Collection

In this project we work with SpaceX launch data that is gathered from an API, specifically the SpaceX REST API. This API give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.

We have the different end points, for example: /capsules and /cores

We work with the endpoint api.spacexdata.com/v4/launches/past.

We use this URL to target a specific endpoint of the API to get past launch data.

We perform a get request using the requests library to obtain the launch data, which we use to get the data from the API.

This result can be viewed by calling the .json() method.

Our response will be in the form of a JSON, specifically a list of JSON objects.

Specifically, we have a list of JSON objects which each represent a launch.

To convert this JSON to a dataframe, we can use the json_normalize function.

This function will allow us to “normalize” the structured json data into a flat table

Another popular data source for obtaining Falcon 9 Launch data is web scraping related Wiki pages.

We use the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records.

Then we need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis.

We transform this raw data into a clean dataset which provides meaningful data on the situation we are trying to address:

Wrangling Data using an API,

Sampling Data, and

Dealing with Nulls.

In some of the columns, like rocket, we have an identification number, not actual data.

This means we need to use the API again targeting another endpoint to gather specific data for each ID number.

We use auxiliary functions that will use the following:

Booster,

Launchpad,

payload, and

core.

The data stored in lists and used to create our dataset.

Another issue we have is that the launch data we have includes data for the Falcon 1 booster whereas we only want falcon 9. We filter the data to remove Falcon 1 launches.

Finally, not all gathered data is perfect. We replace the null values in PayloadMass with the mean of the PayloadMass data. We leave the column LandingPad with NULL values, as it is represented when a landing pad is not used.

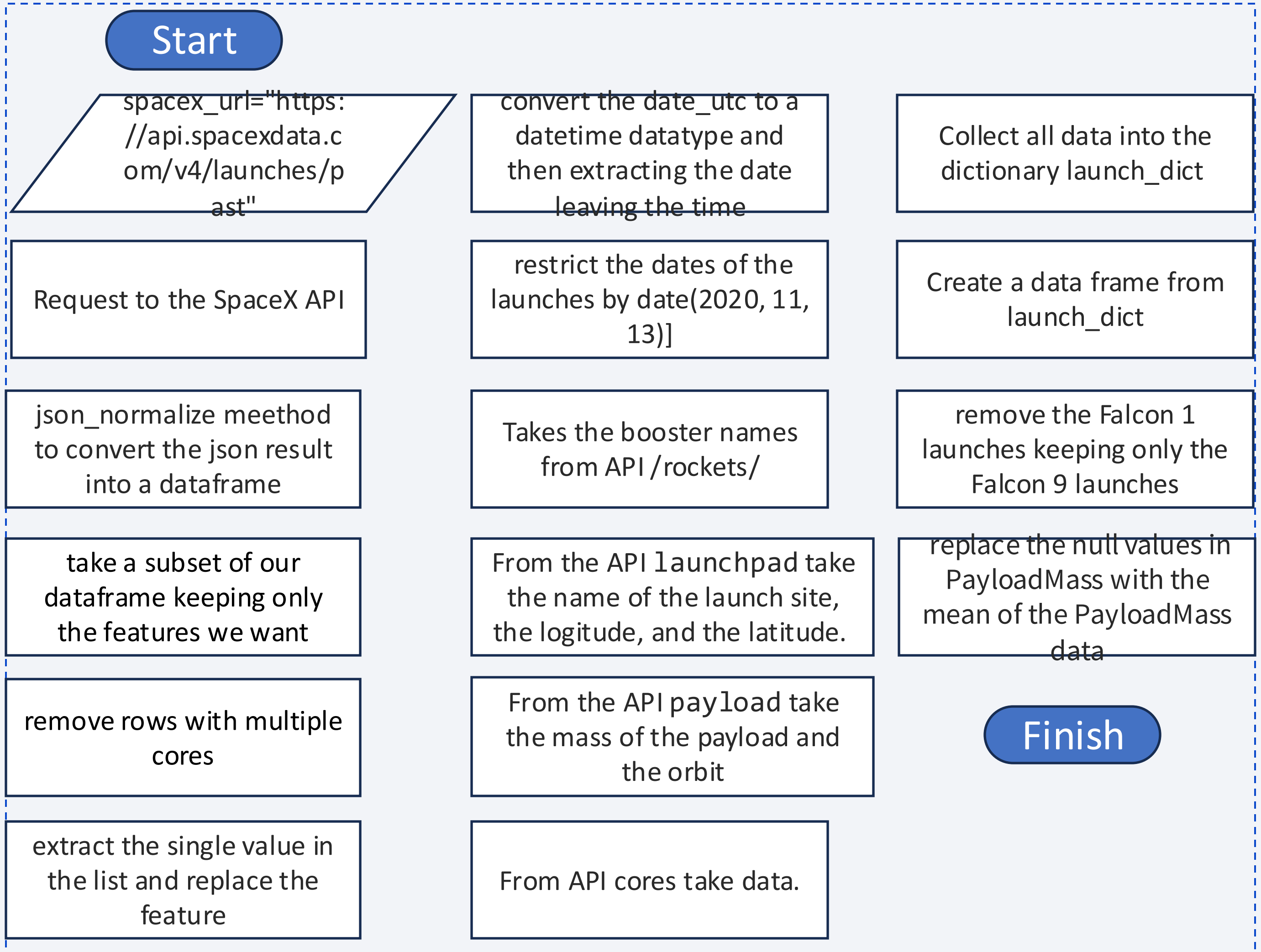
The following Dataframe have been collected.

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5(
1	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5(
2	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Residual stage 1 thrust led to collision between stage 1 and stage 2	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]	5e9e4502f5(

Data Collection – SpaceX API

- SpaceX offers a public API from where data can be obtained

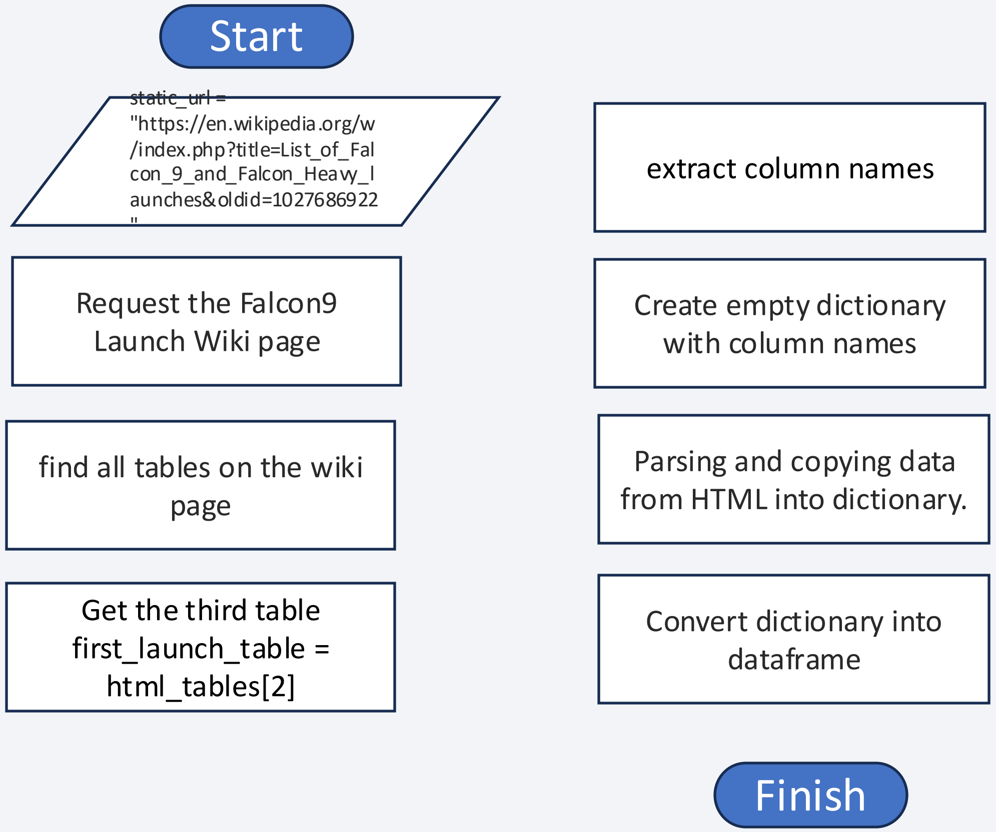
- The completed SpaceX API calls notebook https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Data Collection - Scraping

- WEB scraping to collect Falcon 9 historical launch records from a Wikipedia page titled `List of Falcon 9 and Falcon Heavy launches https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

- WEB scraping notebook https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb



Data Wrangling

In the dataframe there is the column Outcome indicates if the first stage successfully landed.

There are 8 of them:

-True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.

-True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.

-True ASDS means the mission outcome was successfully landed to a drone ship False ASDS means the mission outcome was unsuccessfully landed to a drone ship.

-None ASDS and None None these represent a failure to land.

,We would like landing outcomes to be converted to Classes (either 0 or 1).

- 0 is a bad outcome, that is, the booster did not land.
- 1 is a good outcome, that is, the booster did land.

- The notebook in the GitHub https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Start

Determine the number of `landing_outcomes`
landing_outcomes = df['Outcome'].value_counts()

Create a set of outcomes where the second stage did not land successfully
`{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}`

create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one.

Copy these values into the dataframe
df['Class']=landing_class

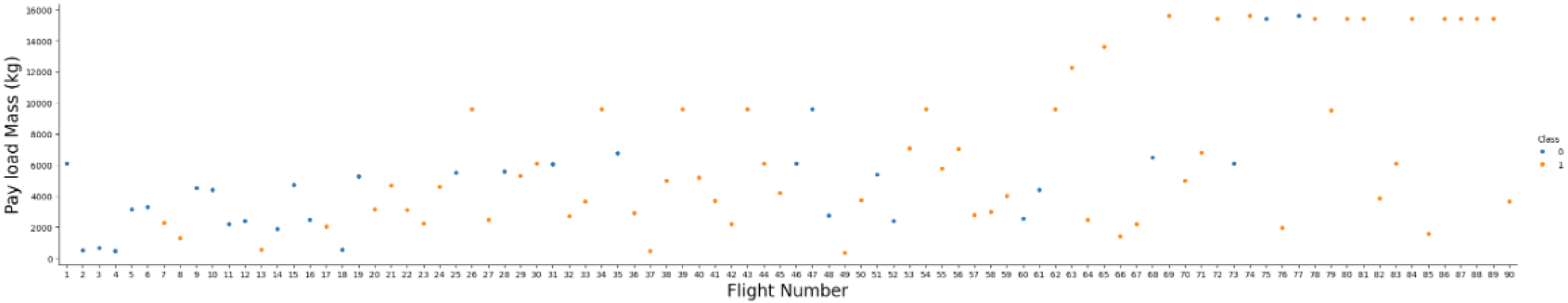
Finish

EDA with Data Visualization

- The following charts were plotted:
  - plot the FlightNumber vs. PayloadMass
  - the relationship between Flight Number and Launch Site
  - the relationship between Payload Mass and Launch Site
  - the relationship between success rate of each orbit type
  - the relationship between FlightNumber and Orbit type
  - the relationship between Payload Mass and Orbit type
  - the launch success yearly trend
- EDA with data visualization notebook https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/edadataviz.ipynb

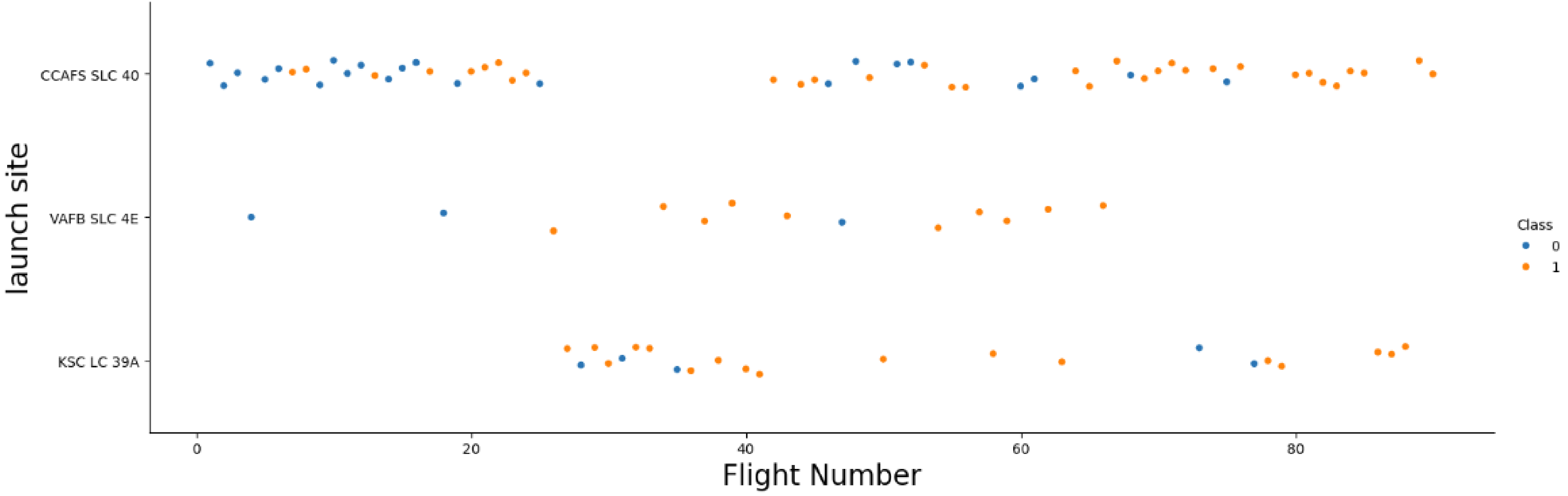
We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass also appears to be a factor; even with more massive payloads, the first stage often returns successfully.

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



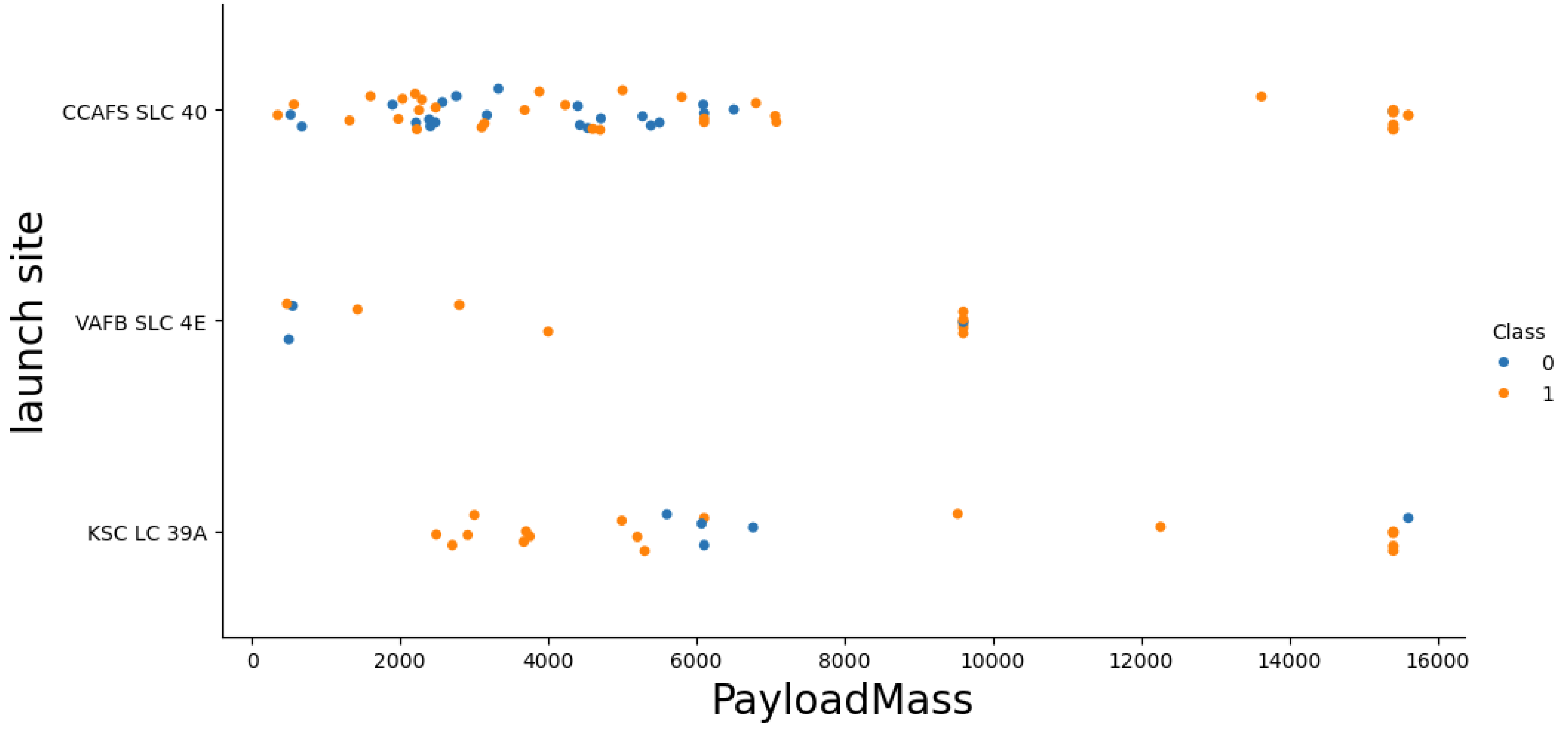
We see that for each site as the flight number increases, the first stage is more likely to land successfully.

```
# 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 = 3)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("launch site",fontsize=20)
plt.show()
```



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

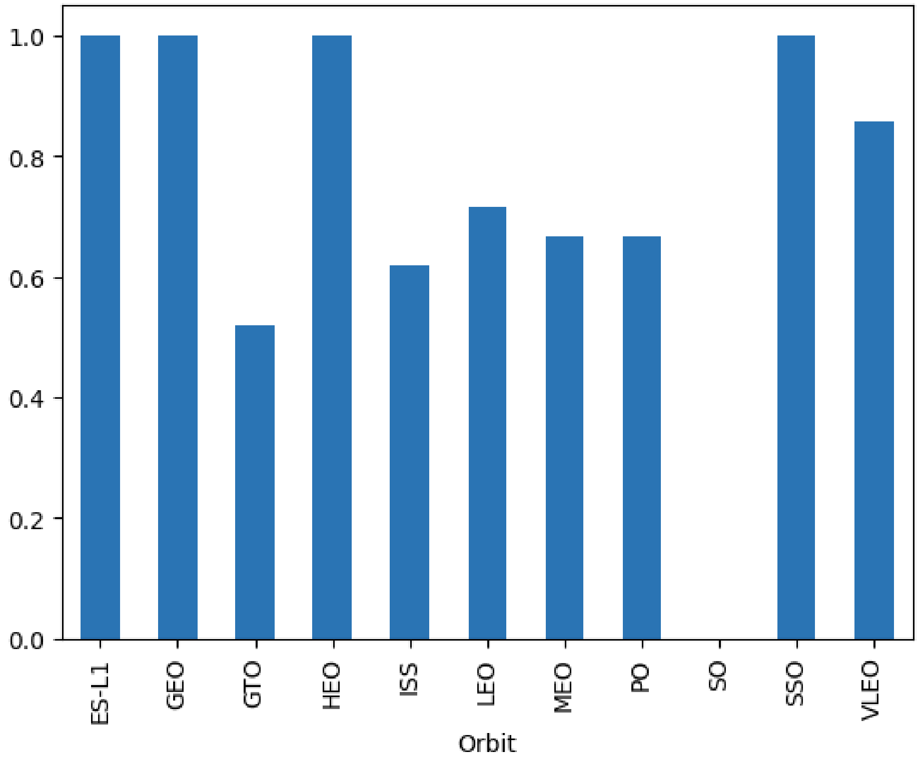
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("launch site",fontsize=20)
plt.show()
```



There are four orbit types for the most success launches: ES-L1, GEO, HEO, SSO.

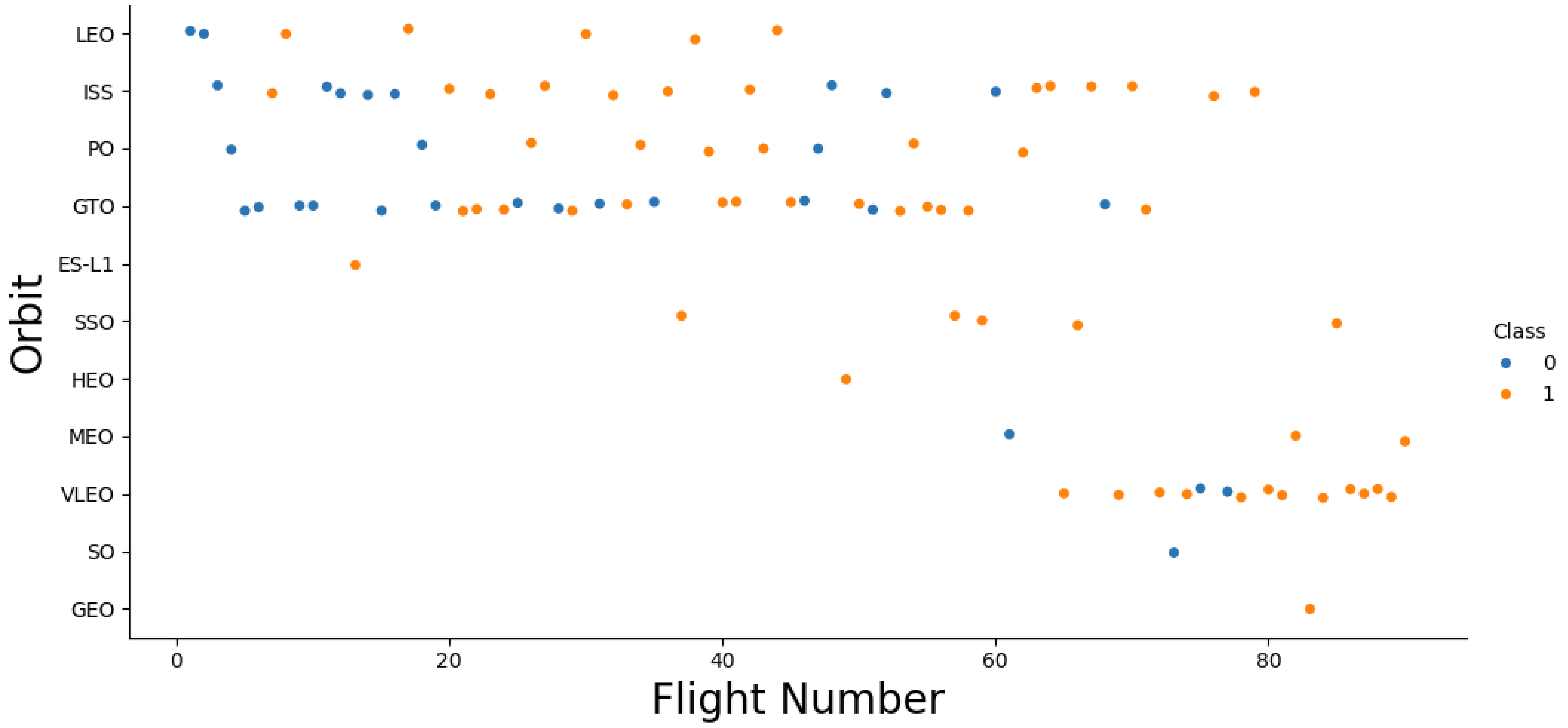
```
df_Orbit = df.groupby('Orbit')['Class'].mean()
df_Orbit.plot.bar(x = 'Orbit', y = 'Class')
```

```
<AxesSubplot:xlabel='Orbit'>
```



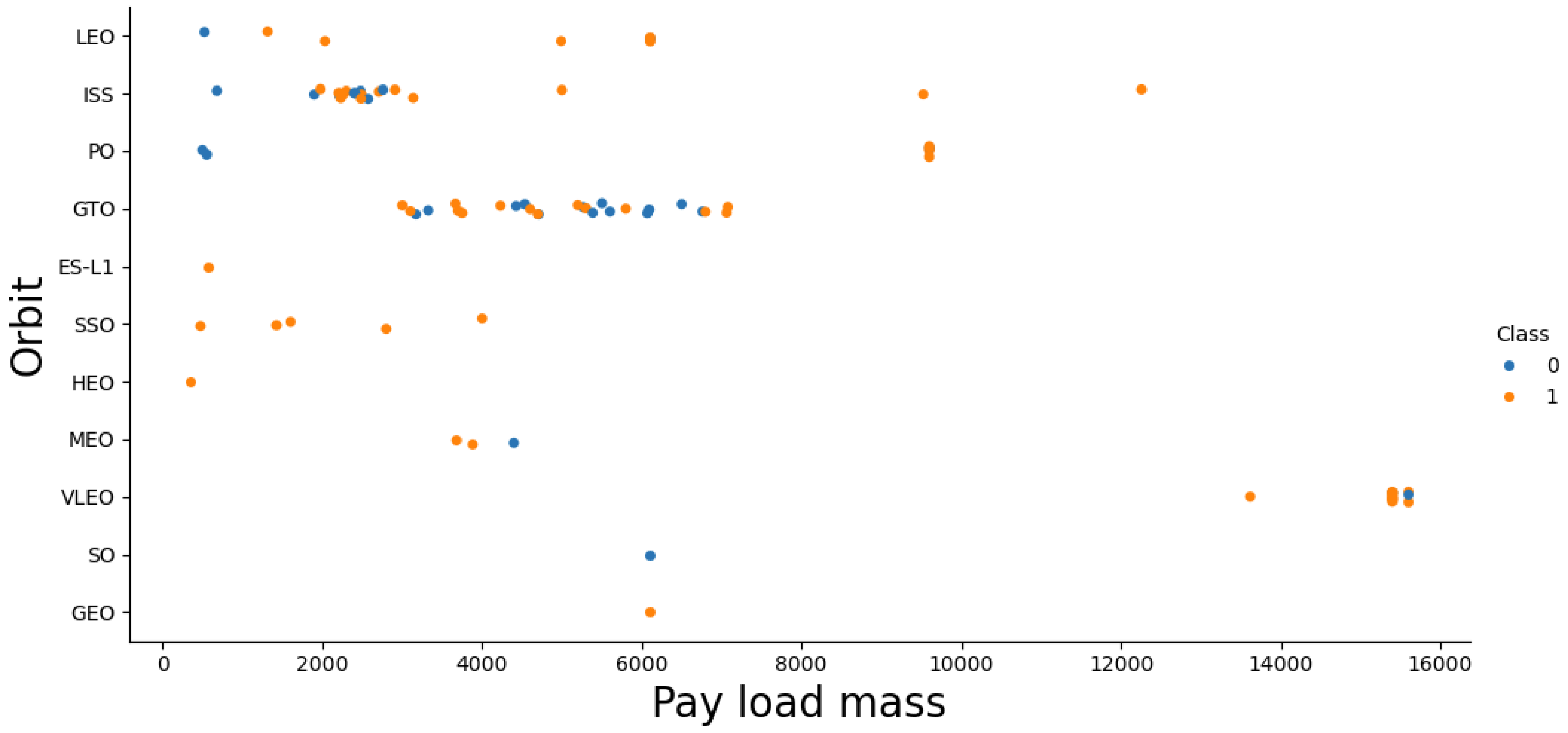
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

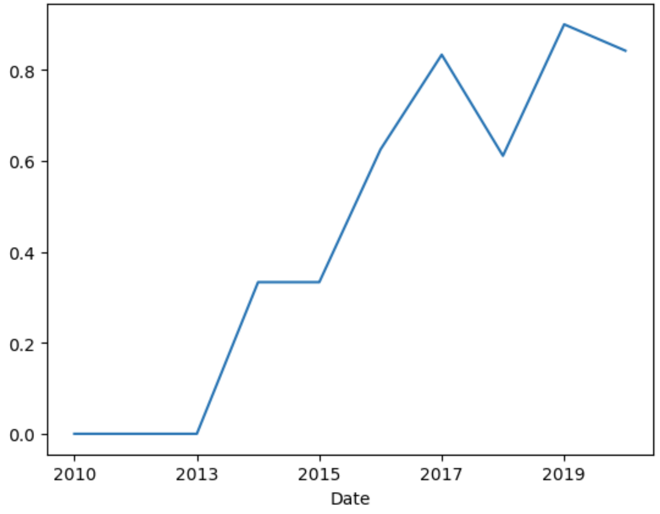
```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("Pay load mass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



A line chart with x axis is the year and y axis is the success rate. You can observe that the sucess rate since 2013 kept increasing till 2020

```python
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_year = df.groupby('Date')['Class'].mean()
print(df_year)
df_year.plot.line(x = 'Date', y = 'Class')
```

```
Date
2010    0.000000
2012    0.000000
2013    0.000000
2014    0.333333
2015    0.333333
2016    0.625000
2017    0.833333
2018    0.611111
2019    0.900000
2020    0.842105
Name: Class, dtype: float64
<AxesSubplot:xlabel='Date'>
```



EDA with SQL

The following SQL queries were perfomed:

- the names of the unique launch sites in the space mission
- 5 records where launch sites begin with the string 'CCA'
- the total payload mass carried by boosters launched by NASA (CRS)
- 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.
- 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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

EDA with SQL notebook https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

- The names of the unique launch sites in the space mission

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

- Records where launch sites begin with the string 'CCA'

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt

- The total payload mass carried by boosters launched by NASA (CRS) = 45596
- Average payload mass carried by booster version F9 v1.1 = 2928.4

- The date when the first succesful landing outcome in ground pad was achieved is 2015-12-22
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

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

- List the total number of successful and failure mission outcomes

Mission_Outcome	count()
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- The names of the booster_versions which have carried the maximum payload mass (15600).

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

- The records which display the month, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

Landing_Outcome	coun
Success	38
No attempt	21
Success (drone ship)	14
Success (ground pad)	9
Failure (drone ship)	5
Controlled (ocean)	5
Failure	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1
No attempt	1

Build an Interactive Map with Folium

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

Build a Dashboard with Plotly Dash

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

Predictive Analysis (Classification)

- We loaded the data using Numpy and Pandas, transformed the data, split data into training testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best perfoming classification model.
- The link to the notebook is https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction.ipynb

Section 3

Launch Sites Proximities Analysis

All launch sites are marked on a map.

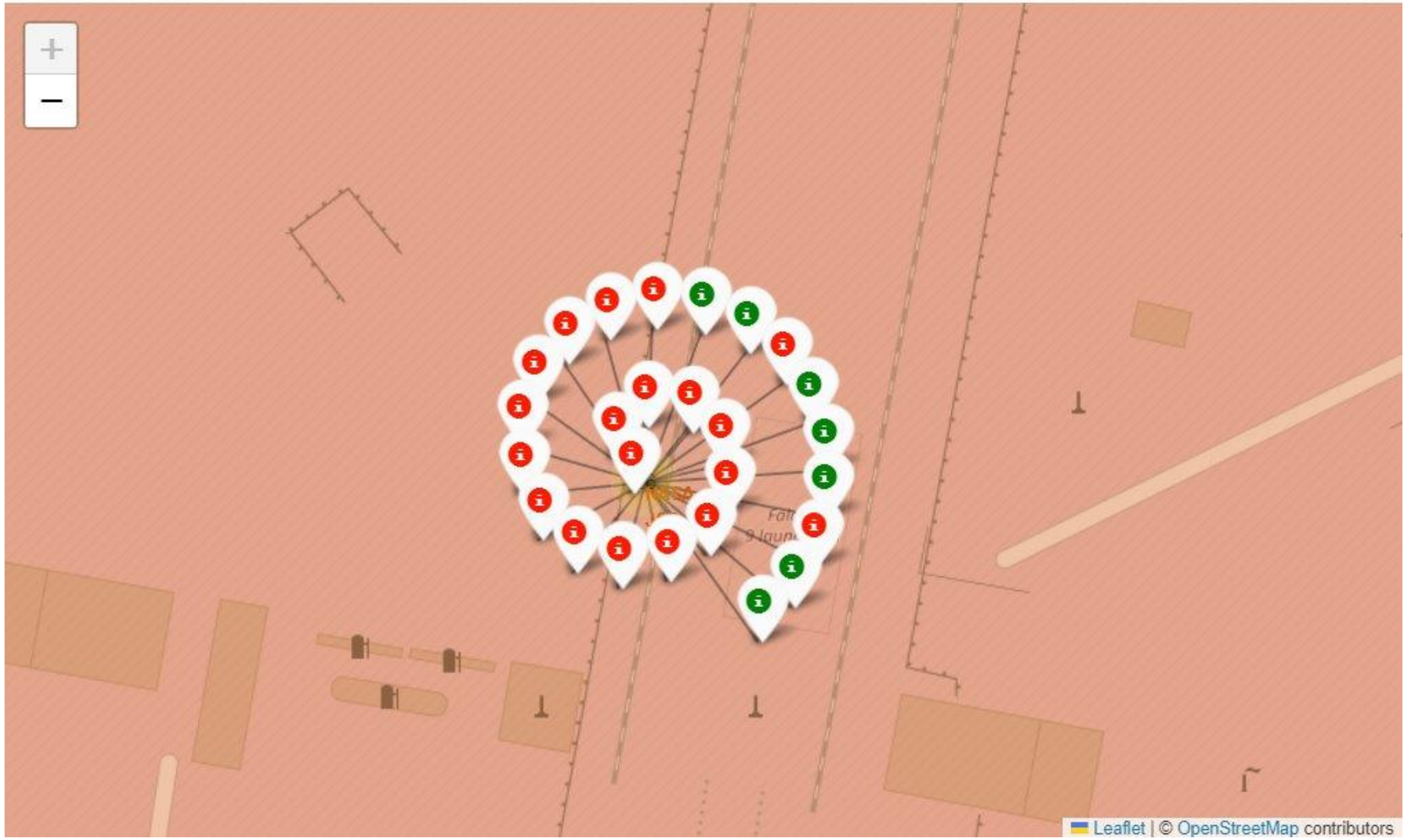
The notebook with built map is https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

We get up coordinates for four launch sites from dataset and put up them on the map. On the map we see three of them are located on East coast USA and next each from other. One is near of West coast.

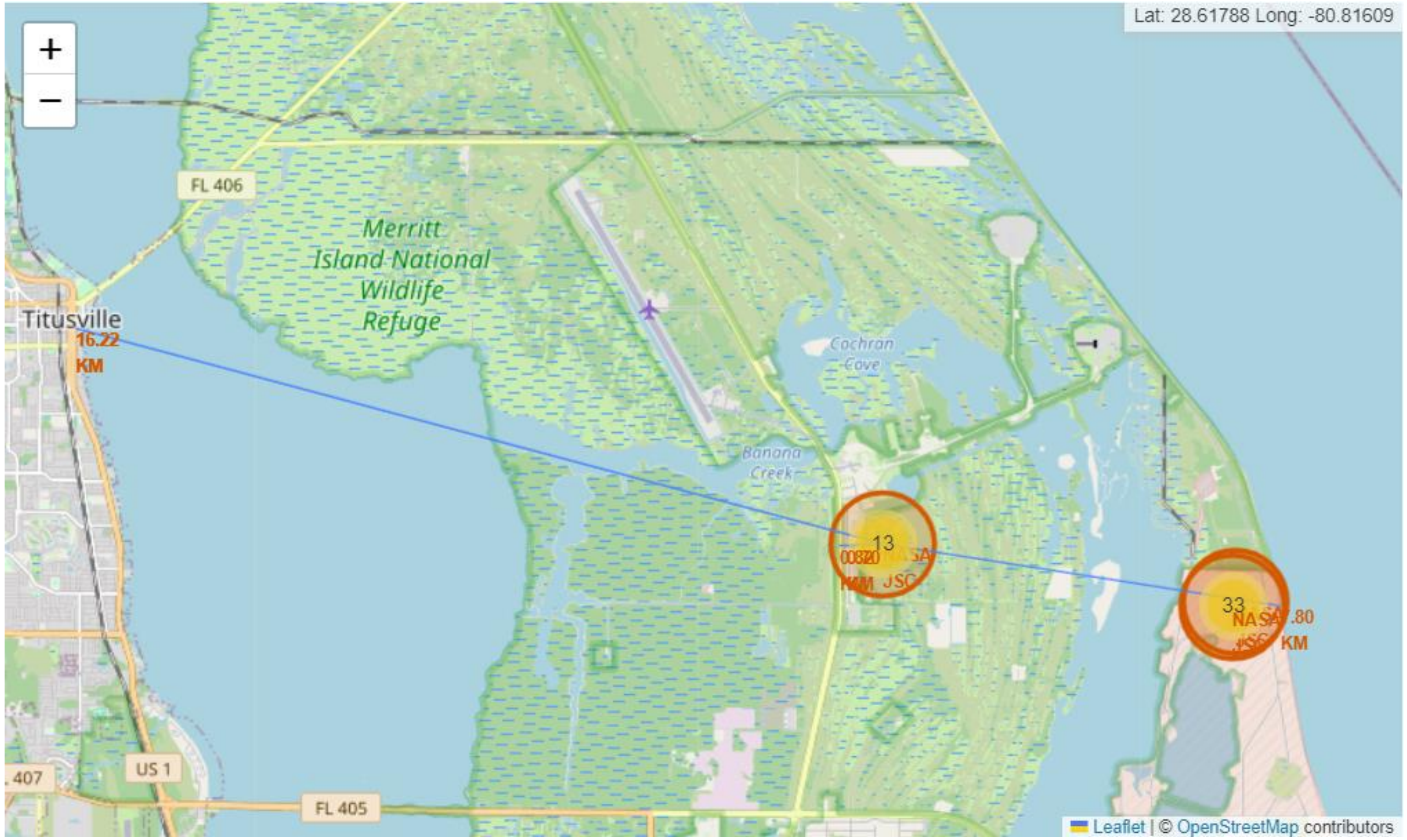
Launch Site	**Lat**	**Long**
CCAFS LC-40	28.562302	-80.577356
CCAFS SLC-40	28.563197	-80.576820
KSC LC-39A	28.573255	-80.646895
VAFB SLC-4E	34.632834	-120.610745

Successes and failed launches on the map.

- The map contains markers that show failed and successes launches. Red markers are failed and green are successes. For instance, on the map below for one of the launch sites we could view more red markers (failed launches) and a few successes.


+
−
Leaflet | © OpenStreetMap contributors

Distances from launch site to a closest city, railway, highway, etc.



- On the map drawed lines from launch sites to some closest objects. The distanses are calculated and showed on the map.
- To the nearest city Titusville 16 km
- To railway 0.7 km
- To highway 0.8 km
- To coastline 7.8 km
- The transport ways are next, this is convience.
- The city is pretty far.

Section 4

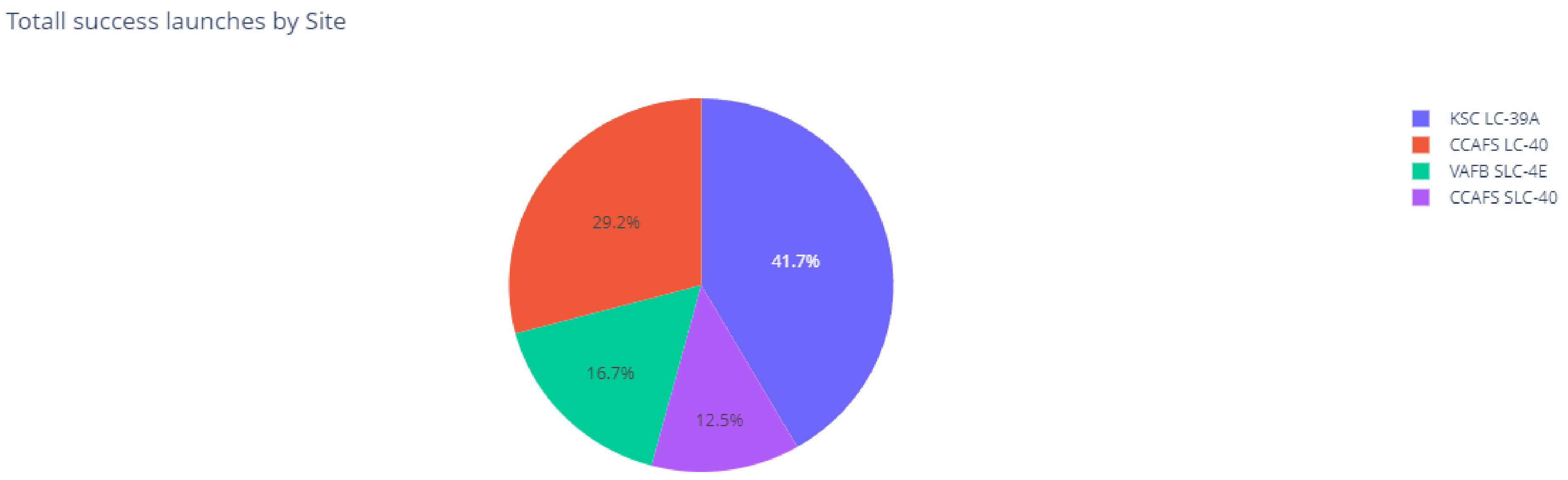
Build a Dashboard with Plotly Dash

The dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

The source Python text https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

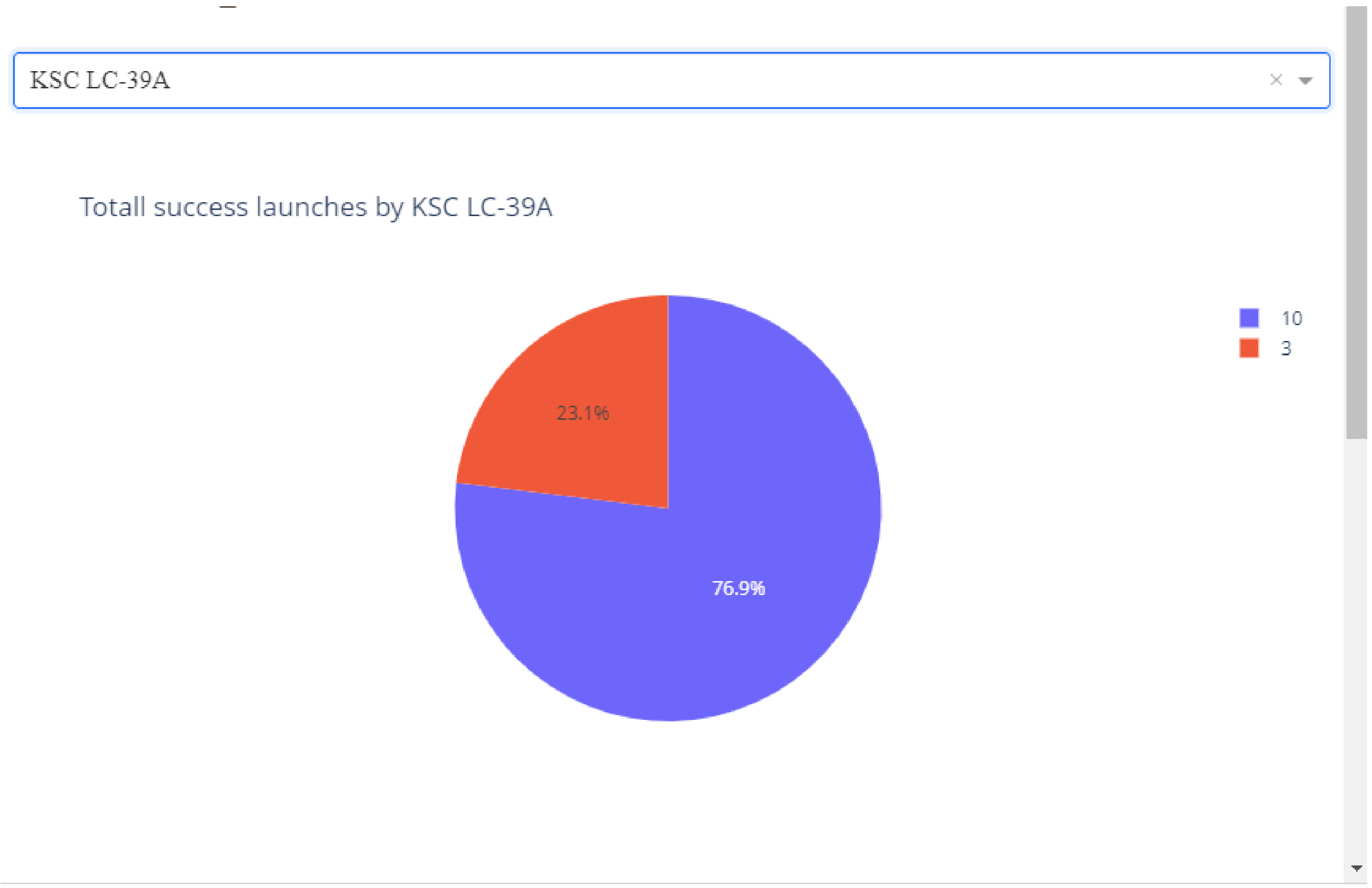
SpaceX Launch Records Dashboard

All Sites



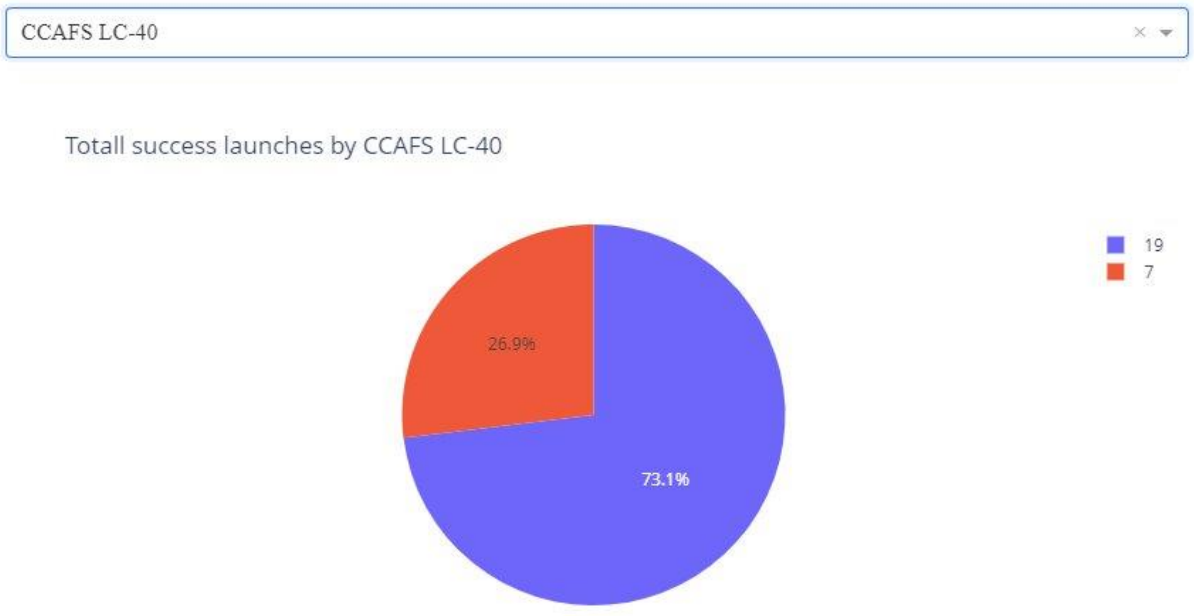
- Piechart shows success launches from each site. We look that the most success launches (41.7%) was from KSC LC-39A. And if put mouse on this pie, we can see 10 success launches (class = 10).

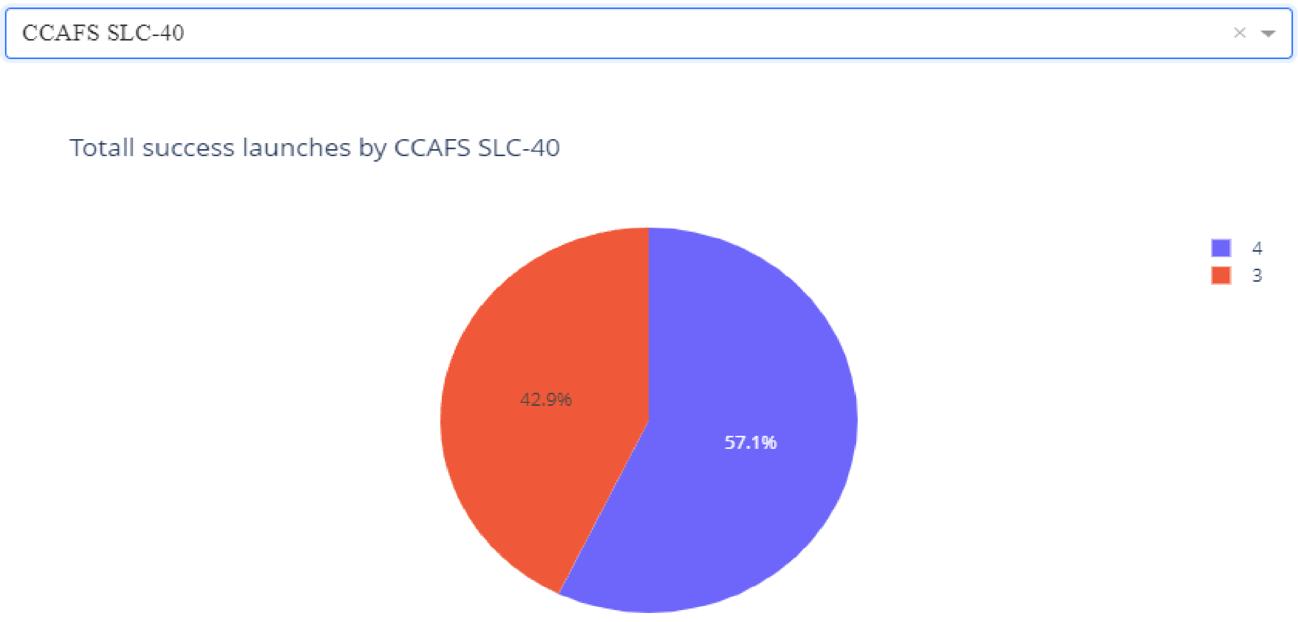
Total success launches for the launch site (KSC LC-39A) with highest launch success ratio

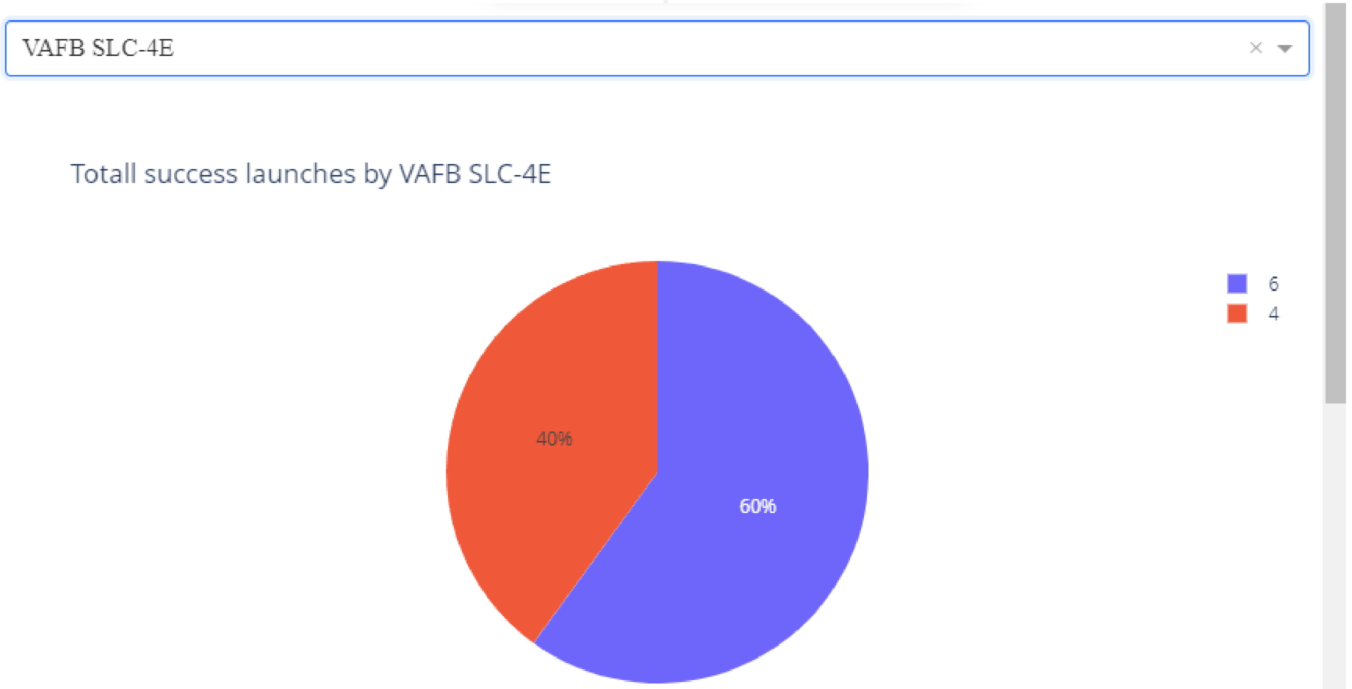


Site KSC LC-39A has 10(76.9%) success launches vs 3(23.1%). This is the most percent among all sites. Below we see percentages for other sites.

CCAFS LC-40 19(73.1%) vs 7(26.9%)
CCAFS SLC-40 4(57.1%) vs 3(42.9%)
VAFB SLC-4E 6(60%) vs 4(40%)


CCAFS LC-40
Totall success launches by CCAFS LC-40
26.9%
73.1%
19
7


CCAFS SLC-40
Totall success launches by CCAFS SLC-40
42.9%
57.1%
4
3


VAFB SLC-4E
Totall success launches by VAFB SLC-4E
40%
60%
6
4

The F9 Booster FT (green color) has the highest success rate.

The range 4600-5300kg has 5 success vs 2 unsuccess and this is the highest success rate = 2.5.

The range 5400-6800kg has 0 success and 6 unseccess launches.

Section 5

Predictive Analysis (Classification)

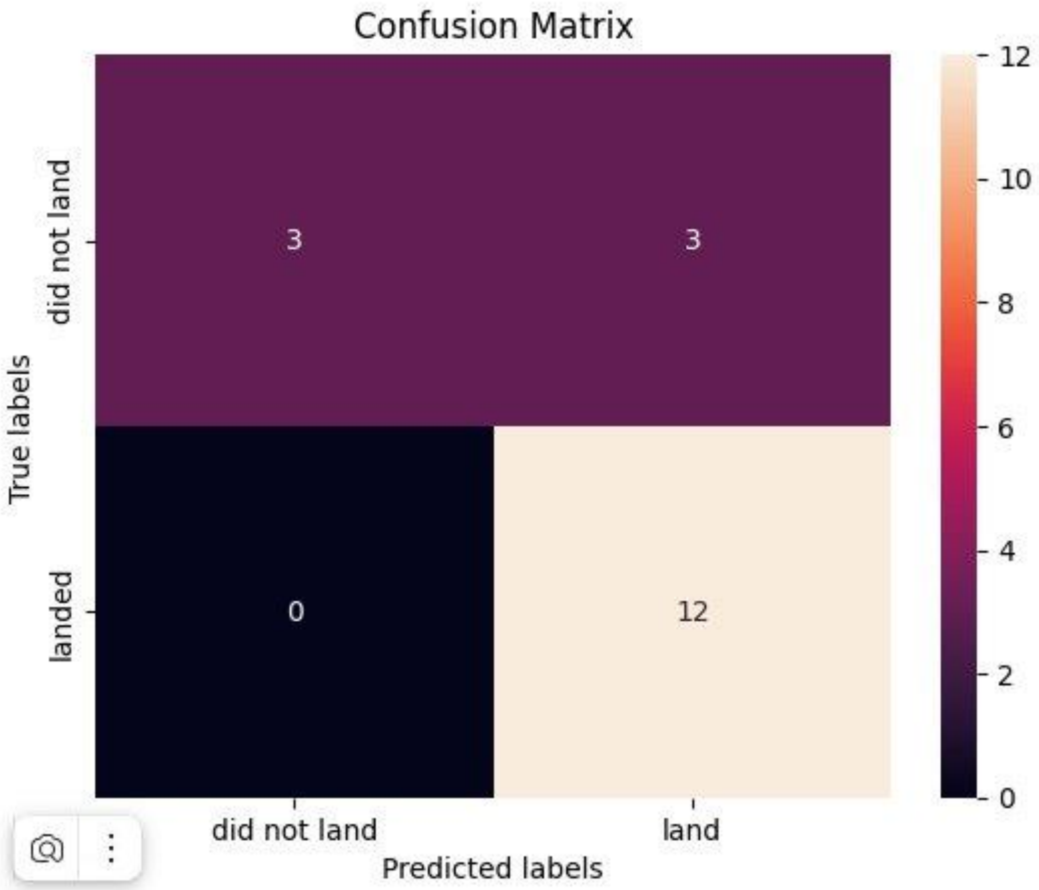
Classification Accuracy

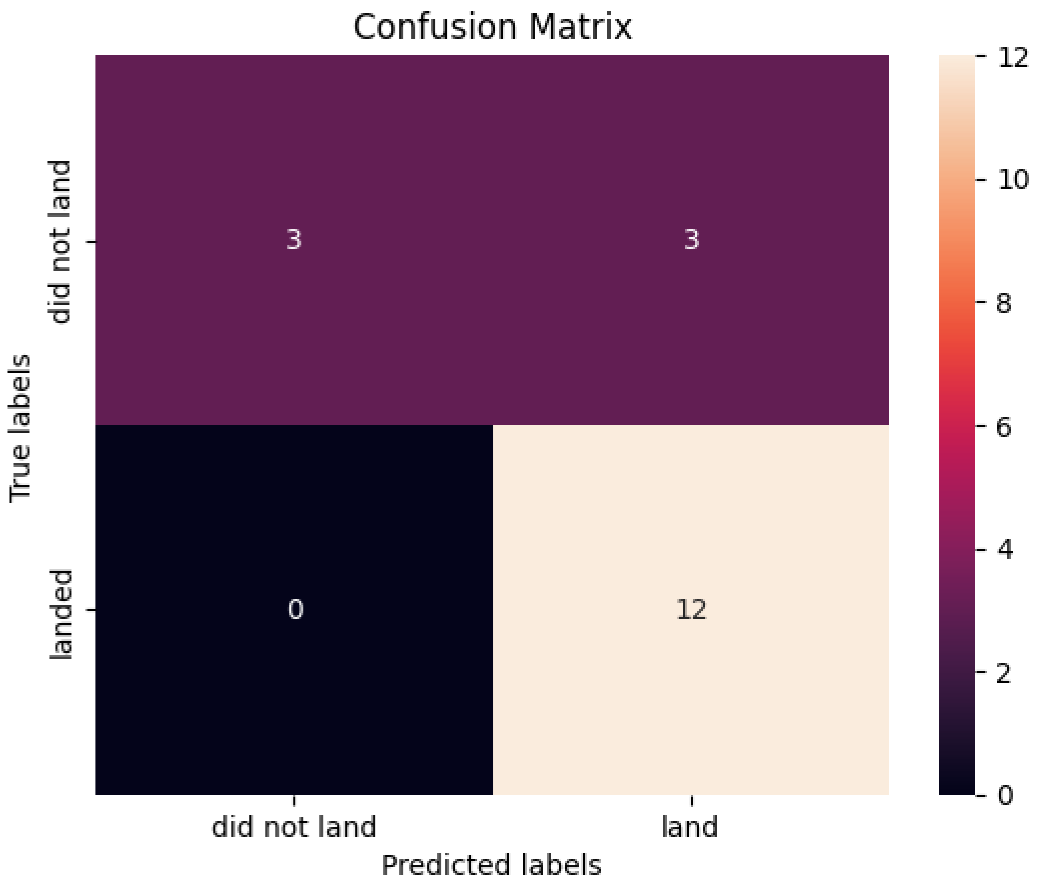
- There are built classification models for four methods.

Method	Logistic_Reg	SVM	Decision Tree	KNN
Test Data Accuracy	0.833333	0.833333	0.888889	0.833333

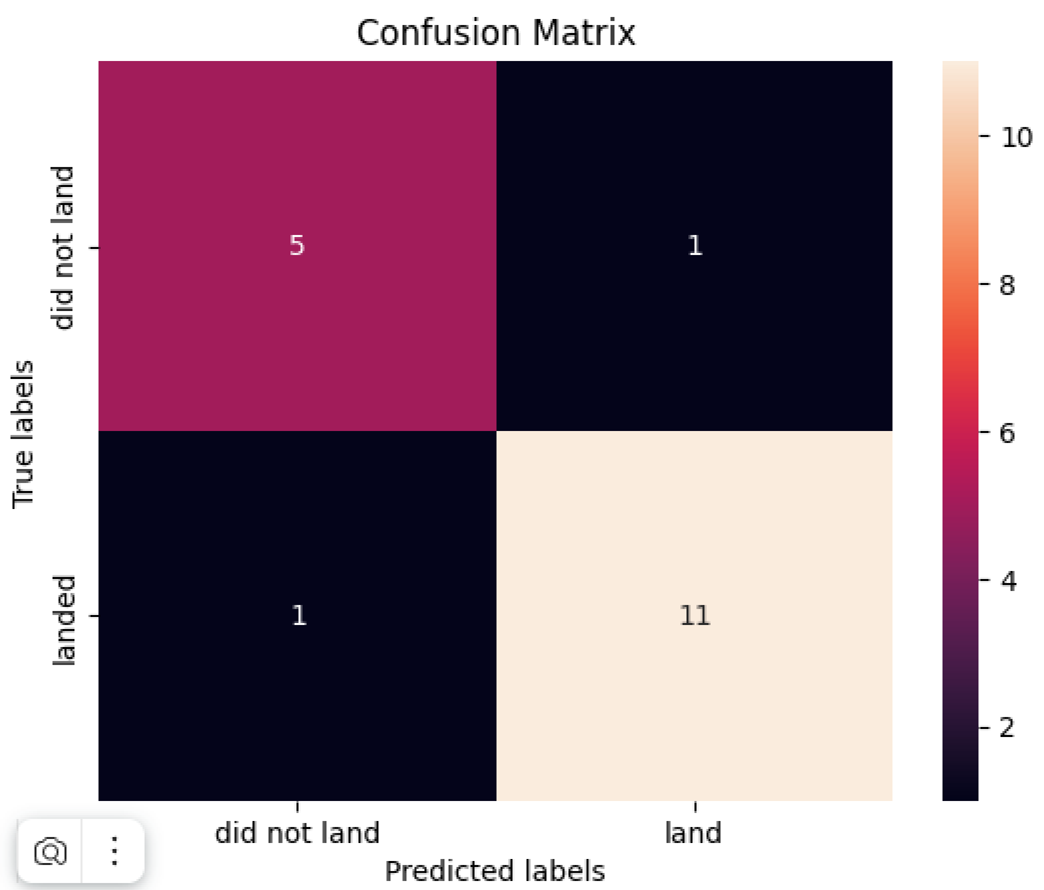
- The highest classification accuracy is 0.89 at Decision Tree Classifier model.
- The notebook is https://github.com/Georgii101/Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction.ipynb

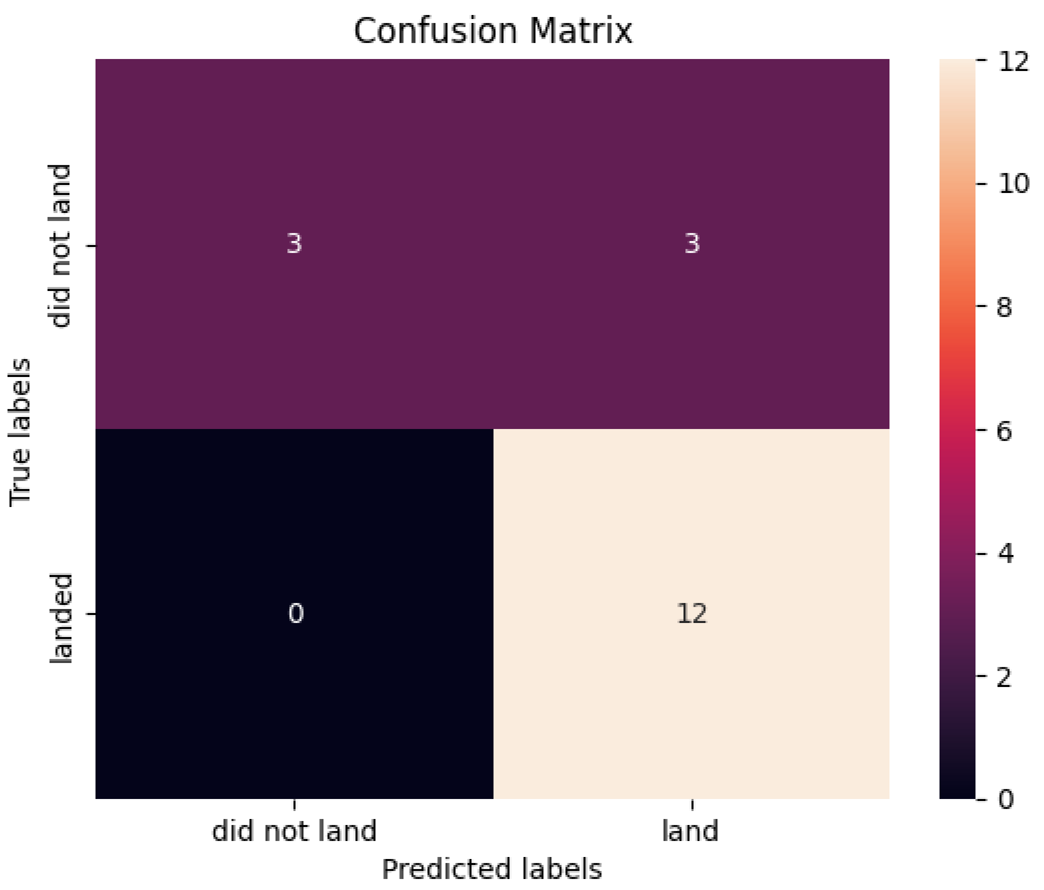
Confusion Matrix





The confusion matrix for logistic regression.	The confusion matrix for SVM.





The confusion matrix for Decision Tree Classifier.	The confusion matrix for k nearest neighbors.

Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

- Assets for Launch Sites Proximities Analysis

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

- Assets for Launch Sites Proximities Analysis

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
```

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

For each launch site, add a Circle object based on its coordinate (Lat, Long) values.

```
for index, site in launch_sites_df.iterrows():
    site_coordinates = [site['Lat'], site['Long']]
    #print(site_coordinates)
    circle = folium.Circle(site_coordinates, radius=1000, color='#d35400', fill=True).add_child(folium.Popup(site['Launch Site']))
    marker = folium.map.Marker(
        site_coordinates,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % site['Launch Site'],
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)
site_map
```

```
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
spacex_df['marker_color'] = spacex_df['class'].apply(lambda x: 'green' if x == 1 else 'red')
```

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)
for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    #marker = folium.Marker()
    site_coordinates = [record['Lat'], record['Long']]
    marker = folium.map.Marker(
        site_coordinates,
        icon=folium.Icon(color = 'white', icon_color = record['marker_color'])
        )
    marker_cluster.add_child(marker)
site_map
```

Create a marker with distance to a closest city, railway, highway, etc. Draw a line between the marker to the launch site

```
city_lat = 28.61113 # Titusville
city_lon = -80.80727
distance = calculate_distance(launch_site_lat, launch_site_lon, city_lat, city_lon)
coordinate = [city_lat, city_lon]
distance_marker = folium.Marker(
    coordinate,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
        )
    )
site_map.add_child(distance_marker)
coordinates = [coordinate,[launch_site_lat,launch_site_lon]]
lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
site_map
```

- # 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):
  if entered_site == 'ALL':
    data = spacex_df
    fig = px.pie(data, values='class',   # берем исходный dataframe, в px.pie указываем разбить по
    names='Launch Site',                  # стартовым площадкам (names='Launch Site')  значения class
                                         # (values = 'class')
    title='Totall success launches by Site')
    return fig
  else:
    # return the outcomes piechart for a selected site
    # отбираем по выбранной стартовой площадке и подсчитываем неудачные и удачные
    #   запуски       (значение class).
    # В легенде показывает количество удачных и неудачных (фиолетовый 6, красный 4),
    # в отличие от приведенного скриншота, где фиолетовый 1, красный 0. Но у меня лучше.
    data = spacex_df[spacex_df["Launch Site"] == entered_site].groupby(['class'])['class'].count()
    fig = px.pie(data, values='class',
    names='class',
    title='Totall success launches by '+entered_site)
    return fig
```

Thank you!