IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

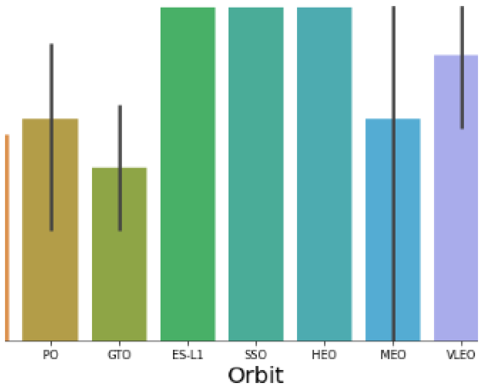
Rafael Fernandes

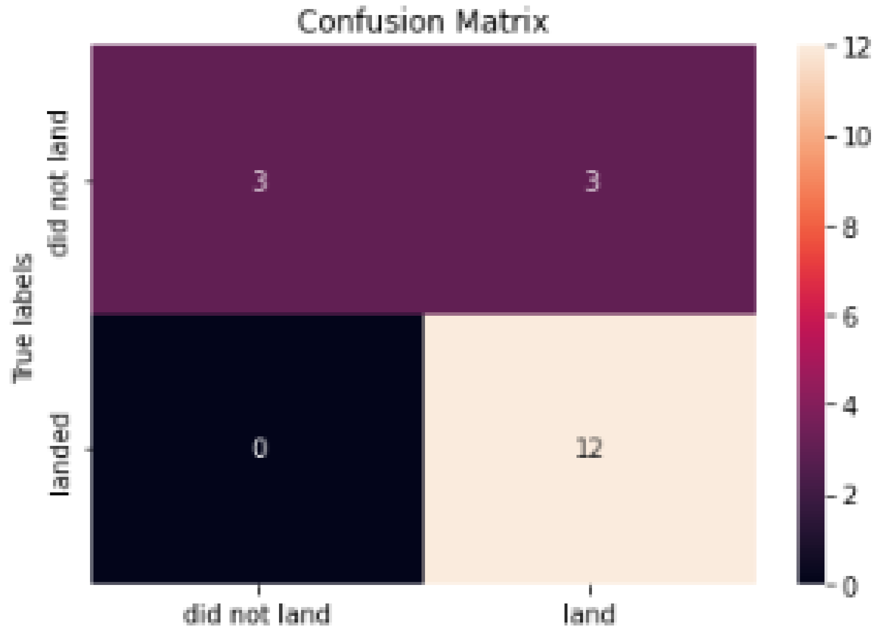
05 May 2022

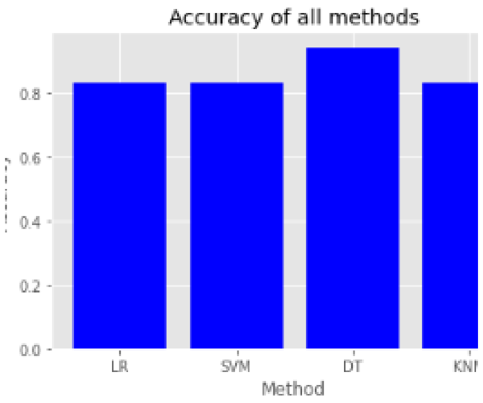
Outline

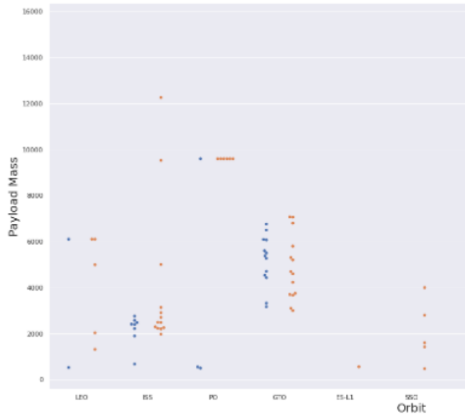
Executive Summary

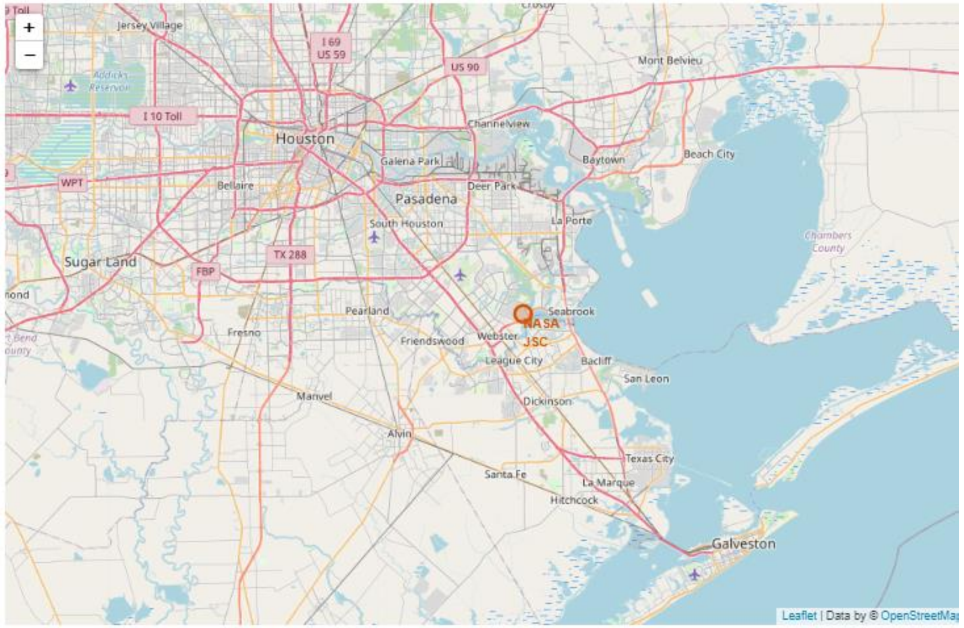
- **Summary of methodologies**
  - Data Collection with SQL, Web Scraping and API
  - Exploratory data and analysis
  - Create Maps with Folium
  - Create Predict Model to analysis the launch Space Y
  - Compared Classification Predict Model to find best
- **Summary of all results**
  - Understanding data with Graphics Visuals and Interactive









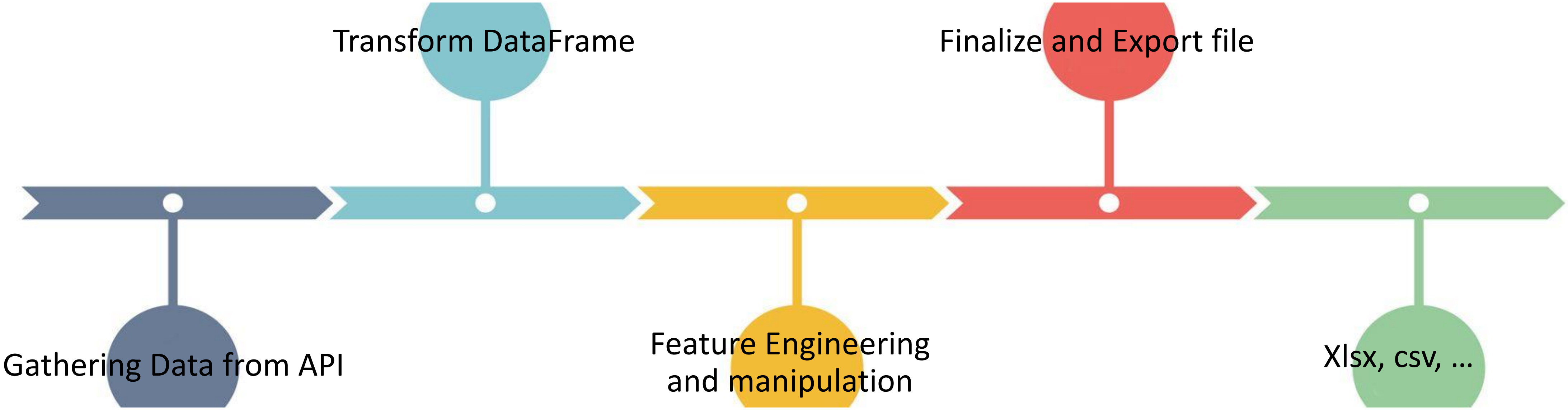


Introduction

- **Project background and context**
  - The idea was to predict if the Falcon 9 will be land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost about of 62 million dollars. Other providers cost upward about of 165 million dollar each, much of the savings is because SpaceX can reuse the first stage. Therefore, we can determine if the first stage will land successfully with bases data historical. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- **Problems you want to find answers**
  - With what factors the rocket will land successfully?
  - The effect of each relationship of rocket variables on outcome.
  - Conditions which will aid SpaceX have to achieve the best results.

Methodology

- **Data Collection:**
  - SpaceX rest API;
  - Web Scraping from Wikipedia;
- **Perform Data Wrangling:**
  - One hot encoding data fields for machine learning and dropping irrelevant columns.
- **Data Analysis using Visualization and Sql:**
  - Scatter, cat plots, bar chart to show patterns between data;
- **Interactive Visual Analytics:**
  - Using Folium maps and Plotly Dash Visualizations;
- **Analysis and Model Predictive using Classification models:**
  - Build and evaluate classification models;



Data Collection

Data collection is the process of gathering and measuring information on targeted variables in an established system which then enables one to answer relevant questions and evaluate outcomes.

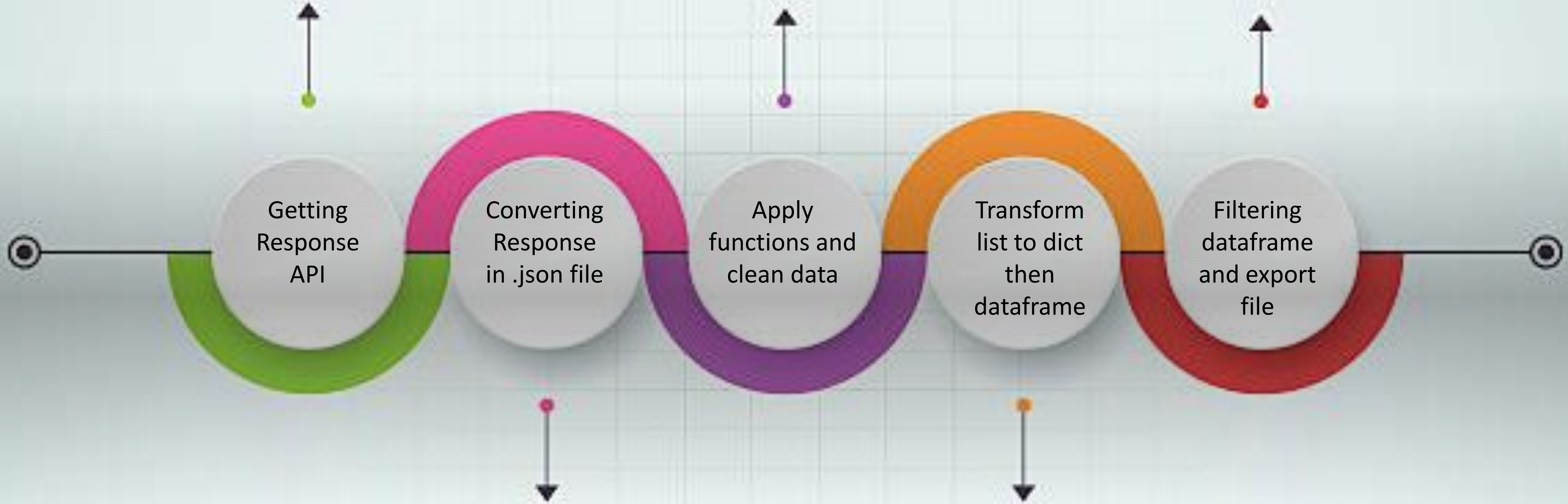
Data Collection – SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

```
BoosterVersion
getBoosterVersion(data)
BoosterVersion[0:5]
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9

data_falcon9.to_csv('dataset_part_1.csv', index=False)
```



GitHub URL

```
jlist = requests.get(static_json_url).json()
df2 = pd.json_normalize(jlist)
df2.head()
```

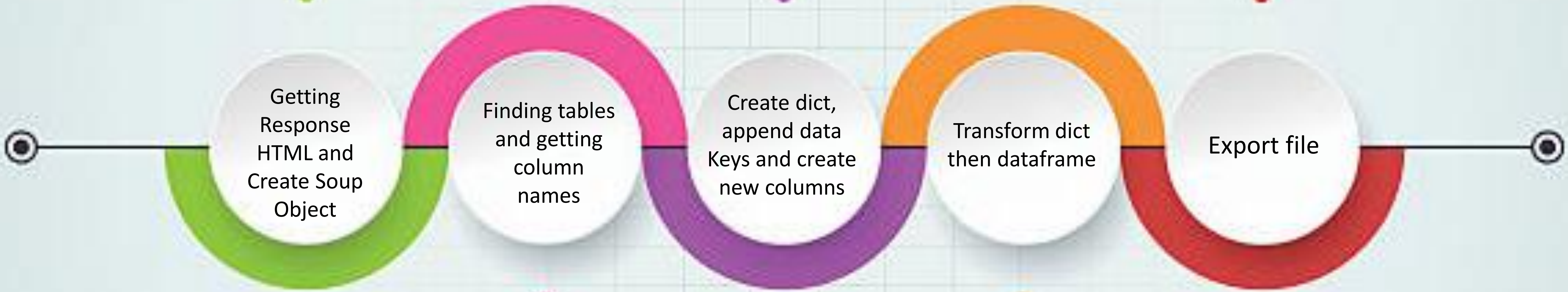
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,'LaunchSite':LaunchSite,
'Outcome':Outcome,'Flights':Flights,
'GridFins':GridFins,'Reused':Reused,
'Legs':Legs,'LandingPad':LandingPad,'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,'Longitude': Longitude,'Latitude': Latitude}

data_falcon9 = pd.DataFrame(launch_dict)
```

Data Collection – Scraping

```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
data  = requests.get(static_url).text
soup = BeautifulSoup(data, 'html5lib')
```

```python
launch_dict= dict.fromkeys(column_names)

del launch_dict['Date and time ( )']

launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []

launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```python
df.to_csv('spacex_web_scraped.csv', index=False)
```




```python
html_tables=soup.find_all("table")
html_tables

first_launch_table = html_tables[2]
print(first_launch_table)

column_names = []
ths = first_launch_table.find_all('th')
for th in ths:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

```python
df=pd.DataFrame(launch_dict)
```

```python
df.head()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10

[GitHub URL]()

Data Collection – Wrangling

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage
art_1.csv")
df.head(10)
```

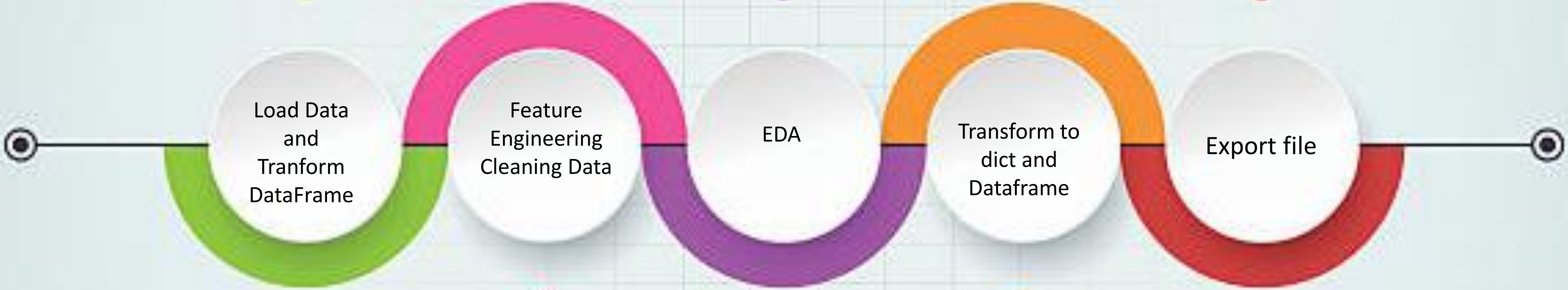
```
# Apply value_counts() on column LaunchSite
df.LaunchSite.value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

```
# Apply value_counts on Orbit column
df.Orbit.value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

```
df.to_csv("dataset_part_2.csv", index=False)
```



GitHub URL

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

We create a set of outcomes where the second stage did not land successfully:

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

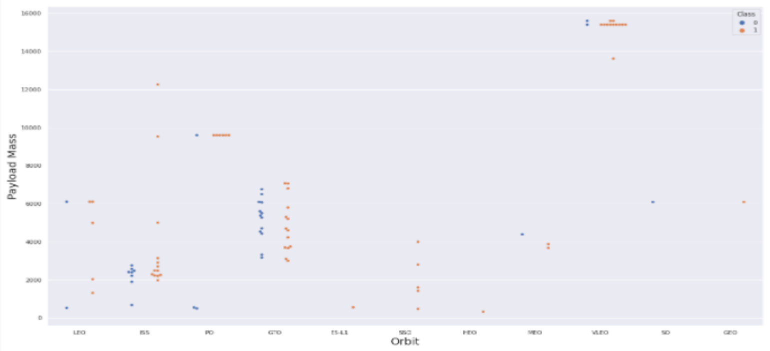
```
df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	Reused
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0

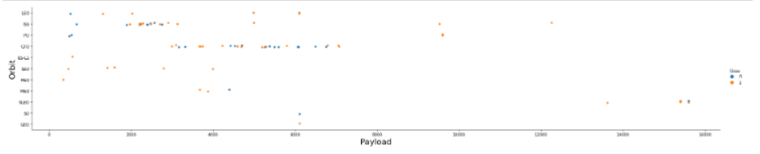
EDA with Data Visualization

- Scatter Graphs Drawn

```
sns.set_theme(color_codes=True)
plt.figure(figsize=(22,12))
sns.swarmplot(x="Orbit", y="PayloadMass", hue="Class", data=df, dodge=True)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Payload Mass",fontsize=20)
plt.show()
```

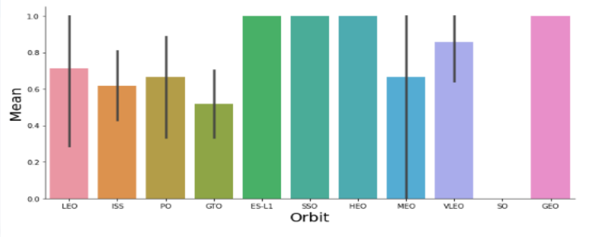



```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value

sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

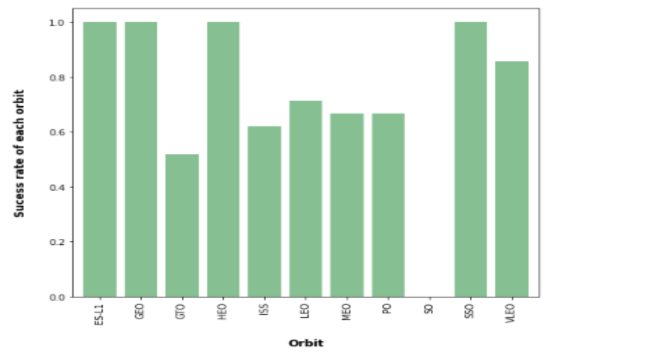


- Bar Graphs Drawn

```
df.groupby(['Orbit']).mean()
sns.catplot(x="Orbit",y="Class", kind="bar",data=df, aspect=2)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Mean",fontsize=20)
plt.show()
```

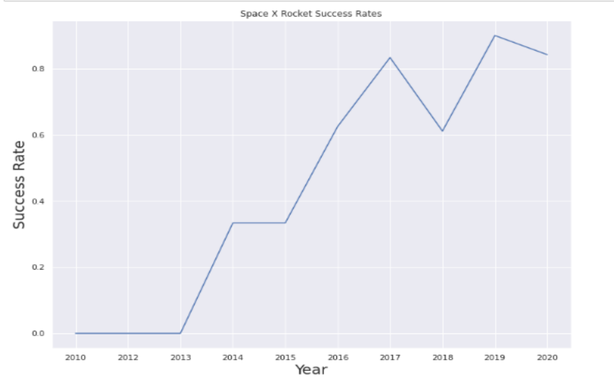



```
xh = df.groupby('Orbit')['Class'].mean()
ax = xh.plot(kind='bar', figsize=(8, 7), color='#86bf91', zorder=2, width=0.8)
ax.set_xlabel("Orbit", labelpad=20, weight='bold', size=12)
ax.set_ylabel("Sucess rate of each orbit", labelpad=20, weight='bold', size=12);
```



- Lines Graphs Drawn

```
df['year']= pd.Series(Extract_year(df["Date"]))
df_groupby_year=df.groupby("year",as_index=False)["Class"].mean()
sns.set(rc={'figure.figsize':(12,9)})
sns.lineplot(data=df_groupby_year, x="year", y="Class" )
plt.xlabel("Year",fontsize=20)
plt.title('Space X Rocket Success Rates')
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



GitHub URL

GitHub URL

- !pip install sqlalchemy==1.3.9
- !pip install ibm_db_sa
- !pip install ipython-sql

- %load_ext sql

- %sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name
-
- %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL

The skills necessary to be a good data scientist include being able to retrieve and work with data, and to do that you need to be well versed in SQL, the standard language for communicating with database systems.

Source: Coursera Site

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 successful 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 failed landing_outcomes in drone ship, their booster versions, and launch site names for 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

Build an Interactive Map with Folium

GitHub URL

Objects Maps	Code	Outcome
Libraries Dash Construction	import folium from folium.plugins import MarkerCluster, MousePosition from folium.features import DivIcon	Use to be performing more interactive visual analytics of way easy. It's possible to shown coordinates and added a Circle Marker around items.
Map Marker	Folium.Marker	Map object to make a mark on map
DivIcon	Folium.Icon	Create an icon on map
Circle Marker	Folium.Circle	Create a Circle where Marker is being placed
PolyLine	Folium.Polyline	Create a line between points
Marker Cluster Object	MarkerCluster	This is a good way to simplify a map containing many markers have the same coordinate
AntPath	Folium.plugins.AntPath	Create na animated line between points

Build a Dashboard with Plotly Dash

GitHub URL

Objects Maps	Code	Outcome
Libraries Dash Construction	import dash import dash_html_components as html import dash_core_components as dcc from dash.dependencies import Input, Output	Plotly is used for Data Viz Interactive graphs. Dash provides all of the available html tags as user-friendly python classes.
Pandas	import pandas as pd	Load file .csv and create dataframe
Plotly	import plotly.express as px	Plot the graphs
Dropdown	dcc.DropDown	Created for launch sites
RangeSlider	dcc.RangeSlider	Created for Payload Mass range selection
Pie Chart	px.pie	Created for Success percentagem display in relation to launch site.
Scatter Chart	px.scatter	Created for correlation display between Payload and Success for all sites or by certain launch site.

Predictive Analysis (Classification)

GitHub URL

Load feature engineered data in dataframe

- Transform it into Numpy arrays
- Standardize and transform data
- Split data into training and test data sets
- Check how many test samples has been created
- List down machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our model

```
Y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
Y_test.shape
```

Finding Best Performing Classification Model

The model with best accuracy score wins the best performing model

```
Logistics Regression method - logreg_cv.score(X_test, Y_test))
Support Vector Machine method - svm_cv.score(X_test, Y_test))
Decision tree method - tree_cv.score(X_test, Y_test))
K nearsdt neighbors method - knn_cv.score(X_test, Y_test))
```

Evaluate Model

- Check accuracy for each model
- Get best hyperparameters for each type of algorithms
- Plot Confusion Matrix

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Results

Exploratory data analysis results

Interactive analytics demo in screenshots

Predictive analysis results

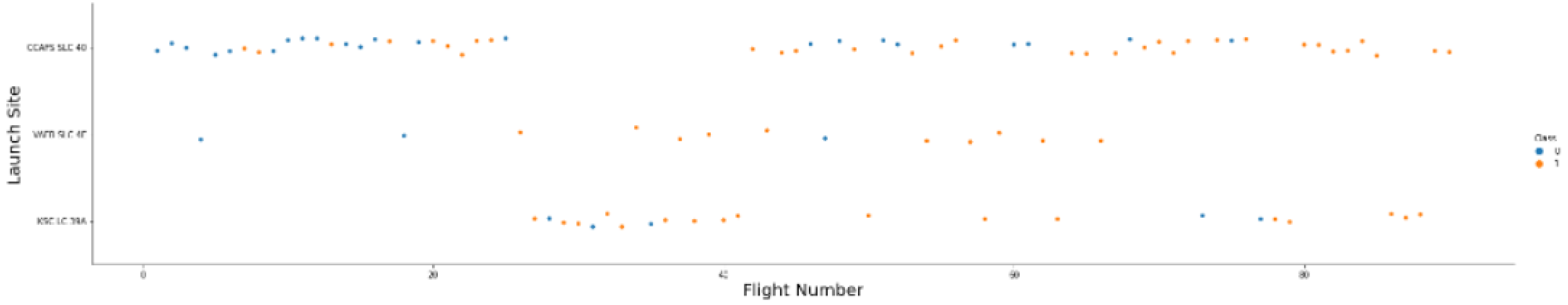
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

With higher flight numbers (greater than 30) the sucess rate for the Rocket is increasing

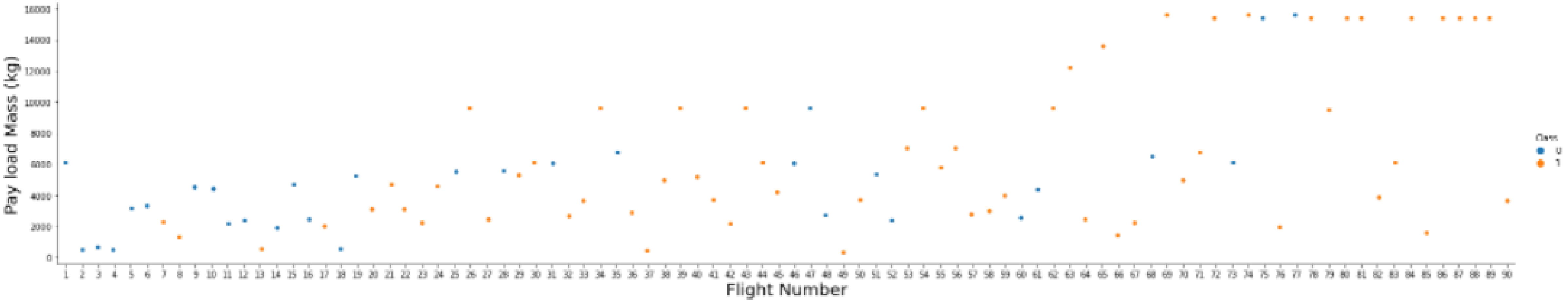
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value

sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class", data=df, aspect = 5)

plt.ylabel("Launch Site",fontsize=20)

plt.xlabel("Flight Number",fontsize=20)

plt.show()
```



Payload vs. Launch Site

The greater the payload mass (greater than 7000kg) higher the success rate for the Rocket but there is no clear pattern to take a decision if the launch site is dependent on PayLoad Mass for a success launch

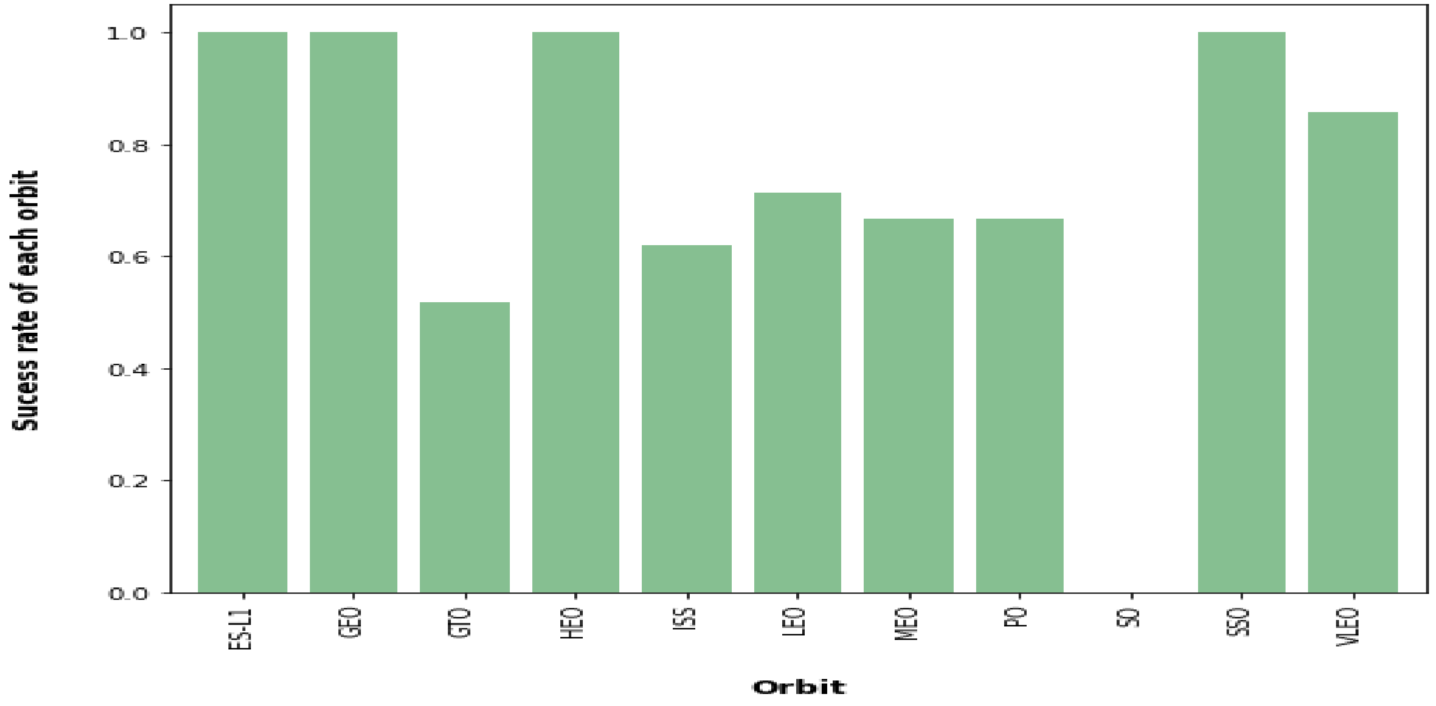
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

ES-L1, GEO, HEO, SSO has highest Success Rates

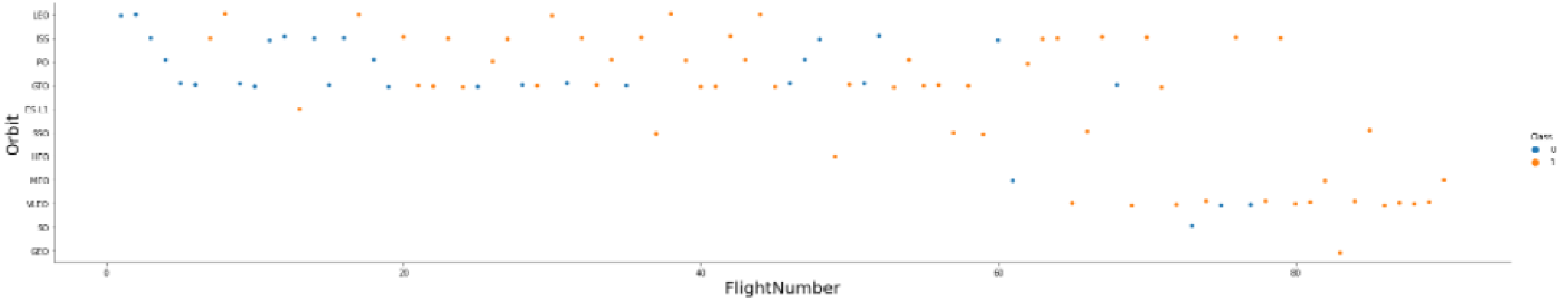
```
xh = df.groupby('Orbit')['Class'].mean()
ax = xh.plot(kind='bar', figsize=(8, 7), color='#86bf91', zorder=2, width=0.8)
ax.set_xlabel("Orbit", labelpad=20, weight='bold', size=12)
ax.set_ylabel("Sucess rate of each orbit", labelpad=20, weight='bold', size=12);
```



Flight Number vs. Orbit Type

We can see that for LEO orbit the success increases with the number of flights
On the other hand there seems to be no relationship between flight number and the GTO orbit

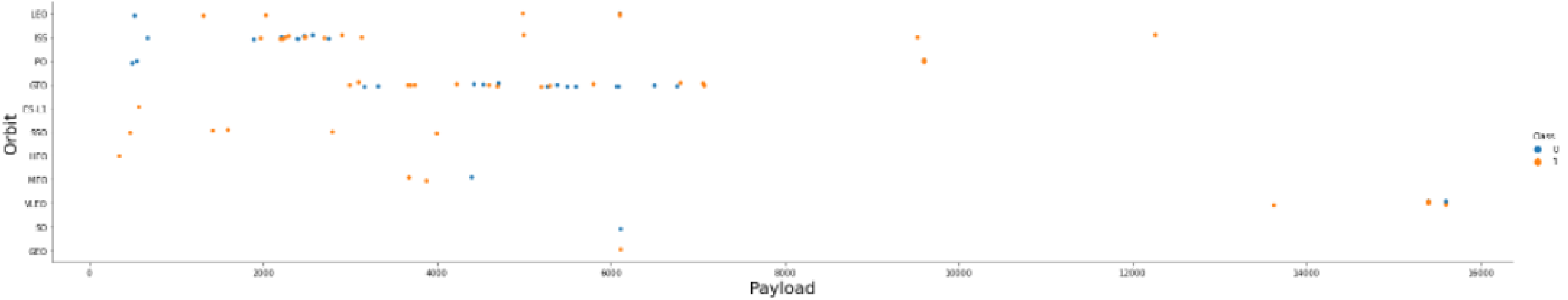
```
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Payload vs. Orbit Type

We can observe that heavy a negative influence on MEO, GTO, VLEO orbits
Positive on LEO, ISS orbits

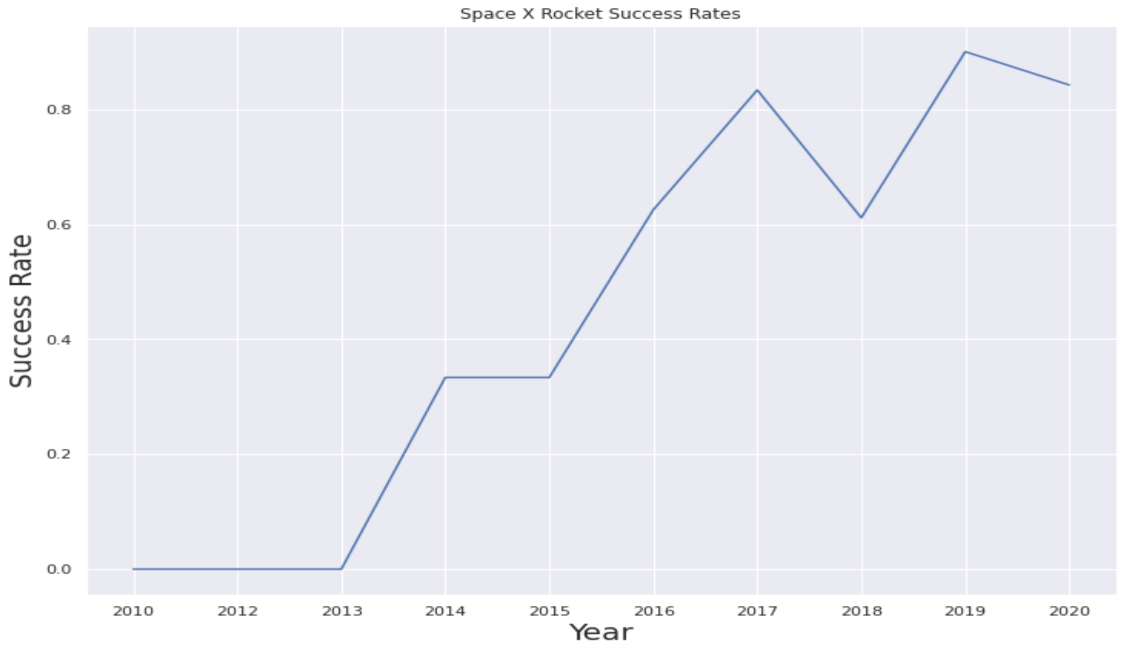
```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

We can observe that the success rate since 2013 kept increasing relatively though there is slight dip after 2019

```
df['year']= pd.Series(Extract_year(df["Date"]))
df_groupby_year=df.groupby("year",as_index=False)["Class"].mean()
sns.set(rc={'figure.figsize':(12,9)})
sns.lineplot(data=df_groupby_year, x="year", y="Class" )
plt.xlabel("Year",fontsize=20)
plt.title('Space X Rocket Success Rates')
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



EDA - SQL

All Launch Site Names

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL
```

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

Using the word DISTINCT in the query we pull unique values for 'launch_site' column from table SPACEX

All Launch Site Names Launch Site Names Begin with 'CCA'

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Using keyword 'limit 5' in the query we fetch 5 records from table SPACEX and with condition 'like' keyword with wild card 'cca%'. The percentage in the end suggests that the 'launch_site' name must start with 'cca'

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

1
45596

Using function SUM calculates the total in the column 'payload_mass_kg_' and 'where' clause filters the data to fetch 'customer' by name 'nasa(crs)'

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1%';
```

1
2928

Using the function 'avg' works out the average in the column 'payload_mass_kg_.
The 'where' clause filters the dataset to only perform calculations on
'booster_version "f9 v1.1“.

First Successful Ground Landing Date

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing__Outcome = 'Success (ground pad)';
```

1
01/05/2017

Using the function 'min' works out the minimium date in the column 'date' and 'where' clause filters the data to only perform calculations on 'landing_outcome' with values 'success (ground pad)'.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Selecting only 'booster_version' 'where' clause filters the dataset to 'landing_outcome = success (drone ship)' 'and' clause specifies additional filter conditions 'payload_mass_kg_ > 4000 and payload_mass_kg_ < 6000'

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Selecting multiple count is a complex query. I have used case clause within subquery for getting both success and failure counts in same query.

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_)FROM SPACEXTBL);
```

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

Using the function 'max' works out the maximum payload in the column 'payload_mass_kg_'

2015 Launch Records

```
%sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE Landing__Outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015;
```

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

We need to list the records which will display the month names, failure 'landing_outcomes' in drone ship, booster versionn, launch_site for the months in year 2015

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

```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE TO_DATE(DATE, 'DDMMYYYY') BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

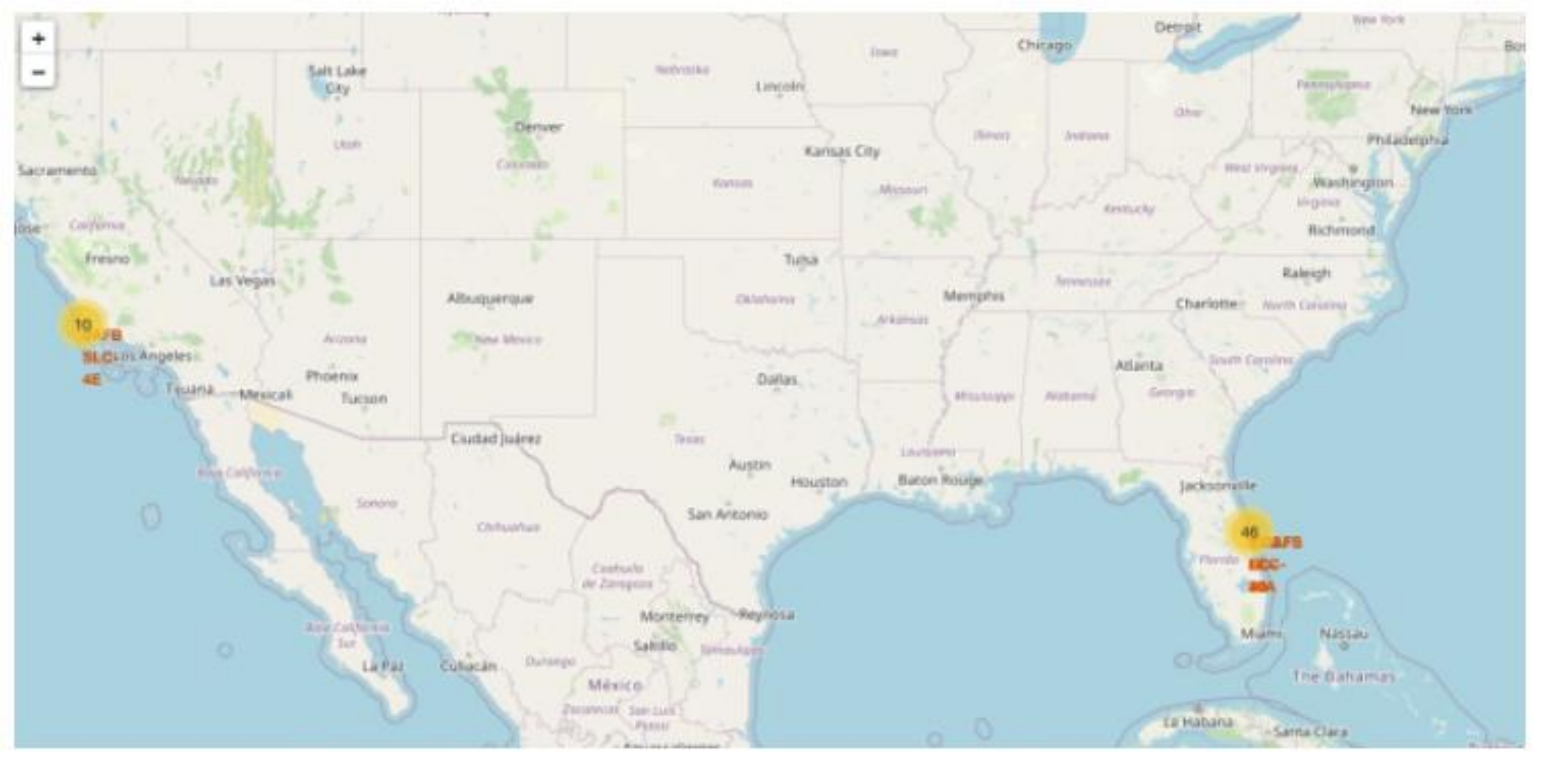
landing__outcome	total_number
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Folium

All launch sites location markers on a global map

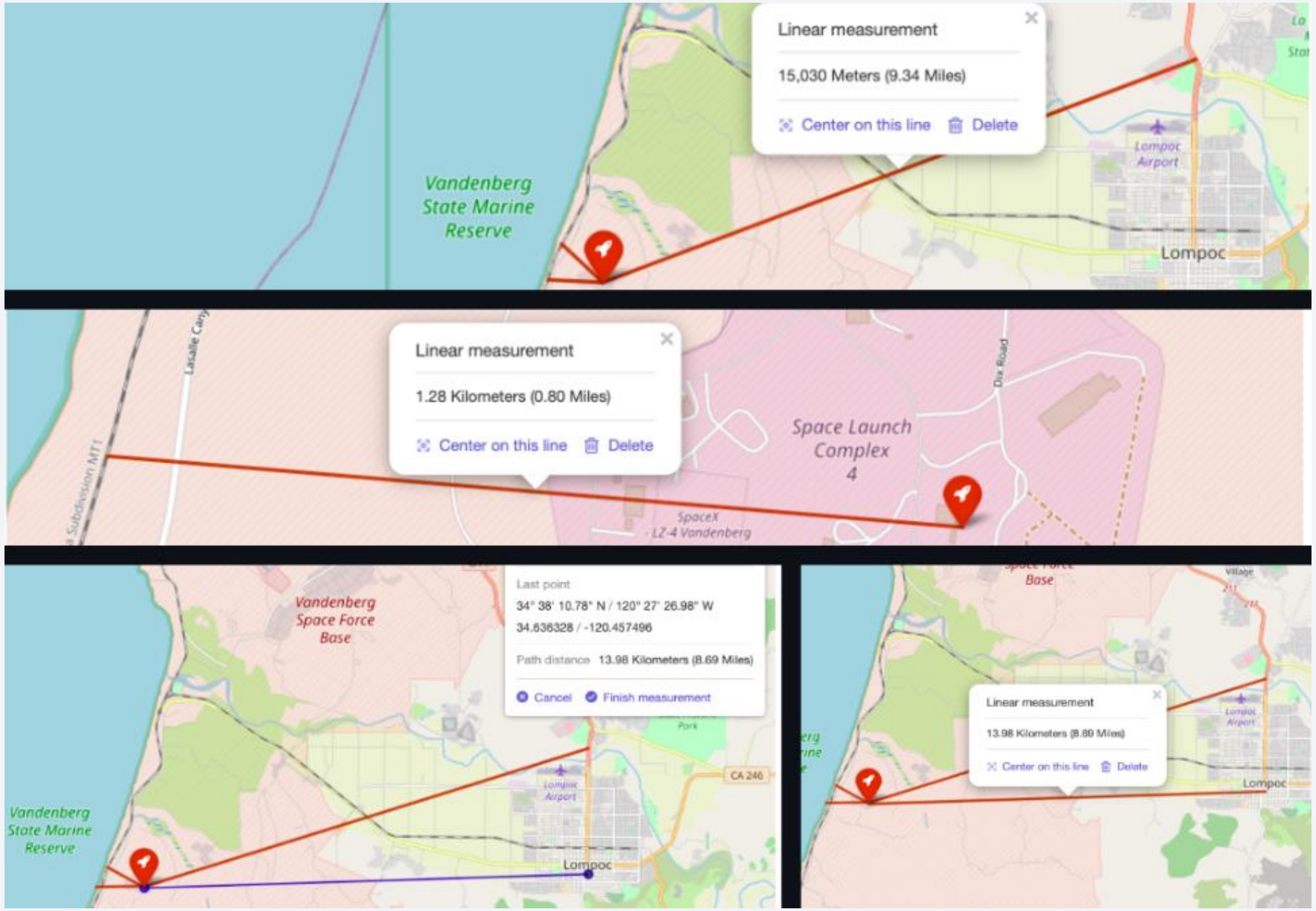


We can that the SpaceX launch sites are near to the USA coasts Floripa and California regions

Color-labeled launch outcomes on the map

Green Marker showns successful launches and Red Marker shows failures

Launch site to its proximities

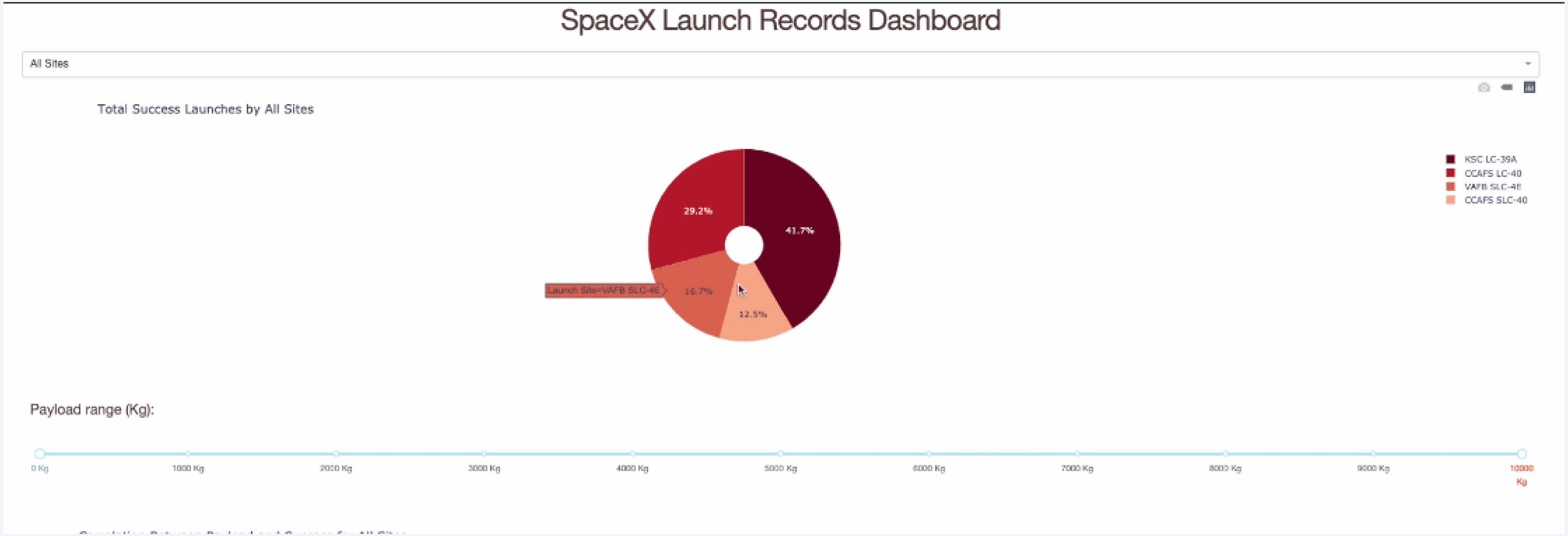


Section 4

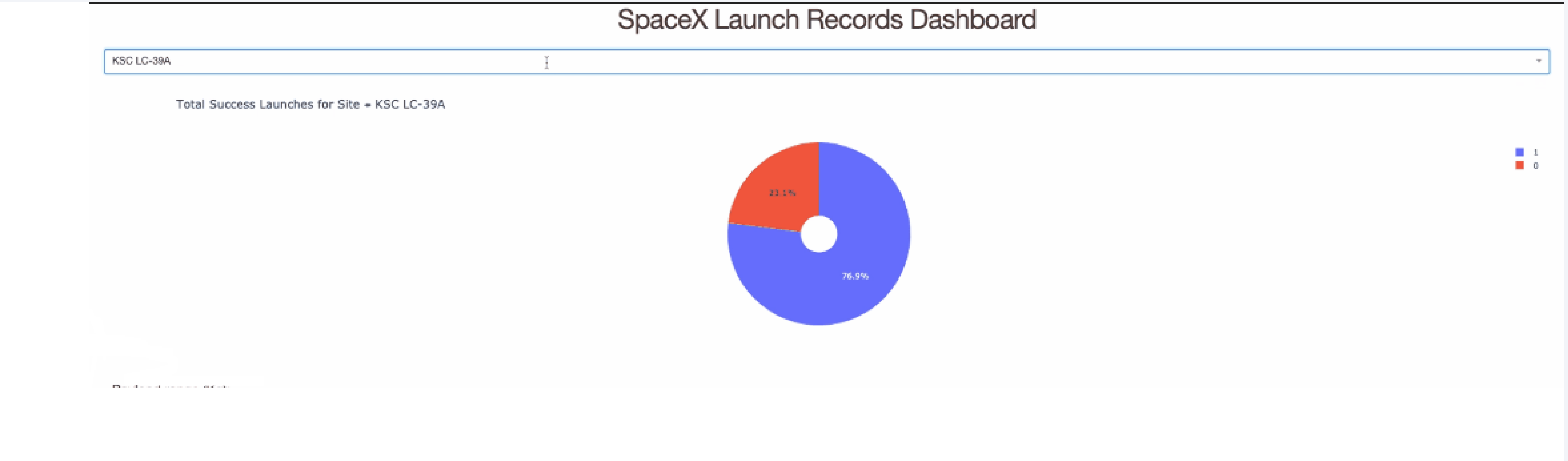
Build a Dashboard with Plotly Dash

Launch Success for All Sites - Count

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



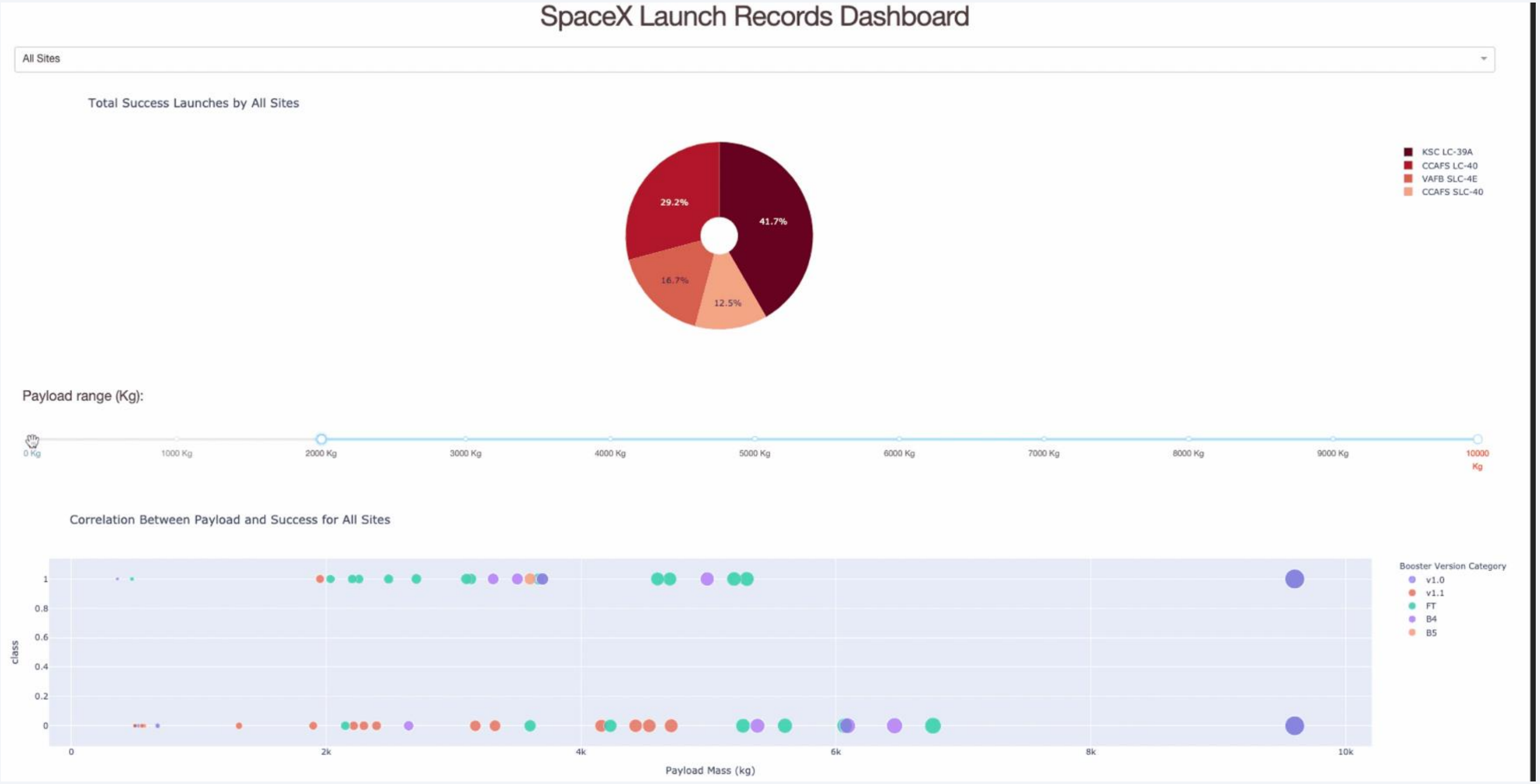
Launch site Highest Success Ratio Launch



KSC LC-39A achieved a 76,9% success rate while getting a 23,1 failure rate

Payload vs. Launch Outcome scatter plot for all sites

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

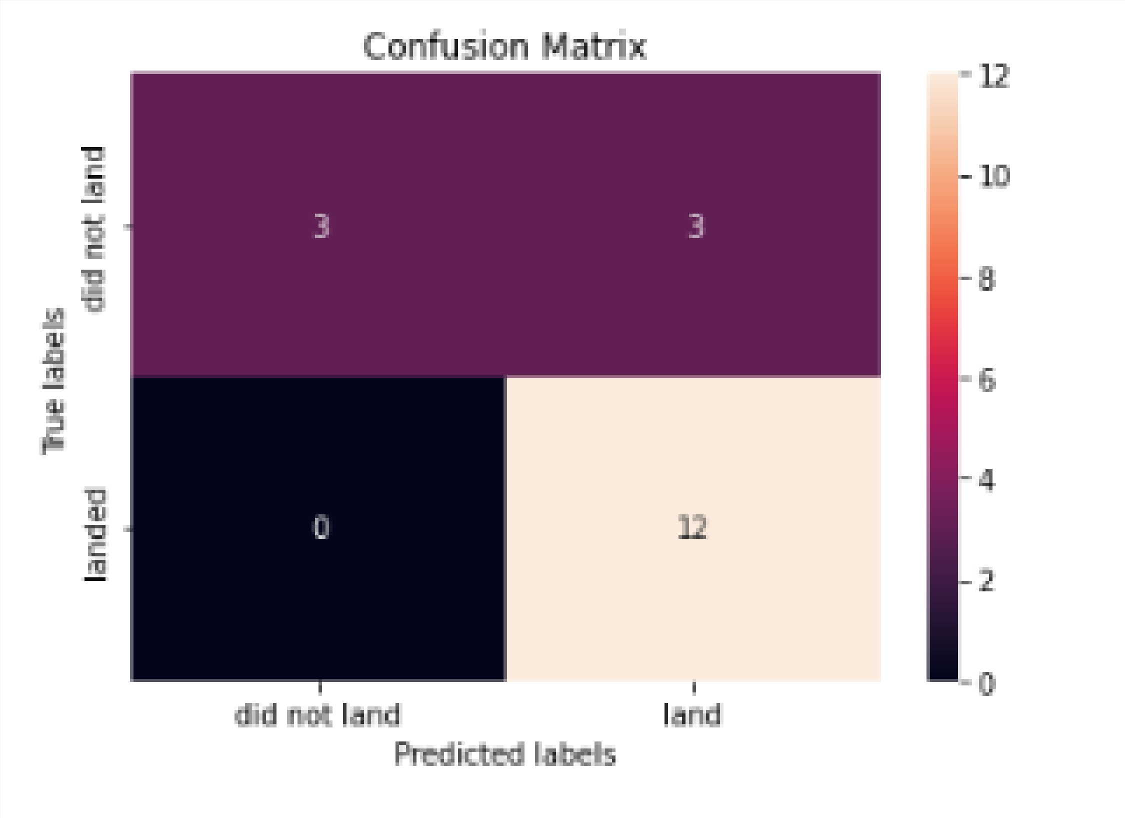


Section 5

Predictive Analysis (Classification)

Classification Accuracy – All models predict

Confusion Matrix – Decision Tree Classifier best score



Conclusions

- Orbits ES-L1, GEO, HEO, SSO has highest success rates
- Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target
- KSC LC-39A had the most successful launches but increasing payload mass seems to have negative impact on success
- Decision Tree Classifier algorithms is the best for machine learning model for provided dataset

Thank you!