'ibhakar

Space X :Capstone Project

Dhaval Vibhakar

27th March 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

Summary of methodologies –

- Data Collection via API, SQL and Web Scraping
- Data Wrangling and Analysis
- Interactive Maps with Folium
- Predictive Analysis for each classification model

Summary.of all results-

- Data Analysis along with Interactive Visualizations

INTRODUCTION

- In this capstone, we will predict if the Falcon 9 first stage will land successfully. 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. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

Problems we 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 S p a c e have to achieve the best results.

METHODOLOGY

Data collection methodology:

- Via SpaceX Rest API
- Web Scrapping fromWikipedia

Perform data wrangling:

- One hot encoding data fields for machine learning and droppingirrelevant columns (Transforming data for Machine Learning)

Perform exploratory data analysis (EDA) using visualization and SQL:

- Using Folium and Plotly Dash Visualizations

Perform predictive analysis using classification models:

- Built 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.
- There are multiple ways to gather data in this project we have used Web mining to Gather our data.

```
data_falcon9
```

```
/tmp/wsuser/ipykernel_164/4023201451.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
```

Out[52]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8

Data Wrangling

- Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.

Load Space X dataset, from last section.

In [2]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

Out[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longit
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577

Identify and calculate the percentage of the missing values in each attribute

In [3]:
```
df.isnull().sum()/df.count()*100
```

Out[3]:
```
FlightNumber     0.000
Date             0.000
BoosterVersion   0.000
PayloadMass      0.000
Orbit            0.000
LaunchSite       0.000
Outcome          0.000
Flights          0.000
GridFins         0.000
Reused           0.000
Legs             0.000
LandingPad      40.625
Block            0.000
ReusedCount      0.000
```

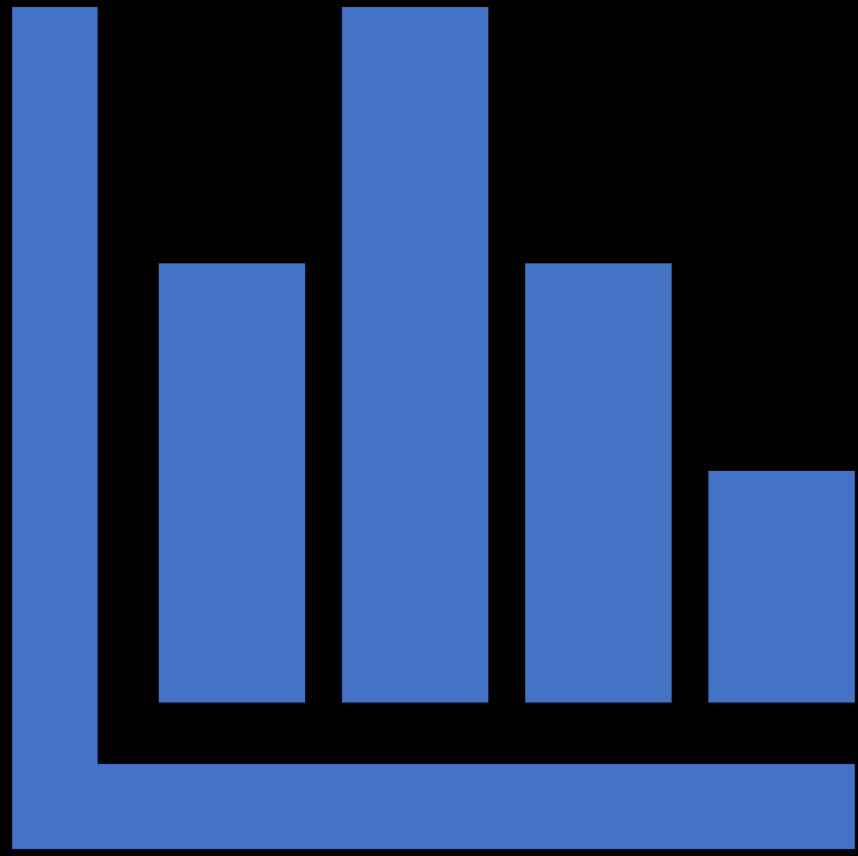
EDA - Meaning & Basic Steps

- Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, using statistical graphics and other data visualization methods.
- Load Data
- Create Data Frame
- Create Visualization
- Collect Insight

EDA with Data Visualization

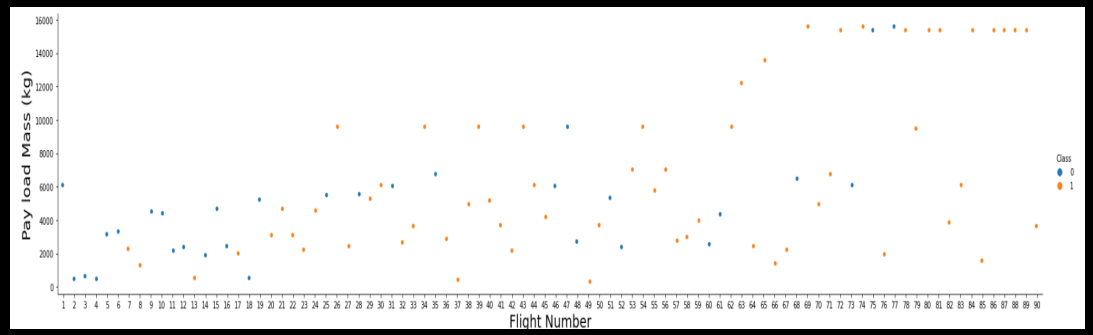
Scatter Graphs Drawn:

1. Payload and Flight Number
2. Flight Number and Launch Site
3. Pavload and Launch Site Flight Number and Orbit Type Payload and Orbit Type
4. Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs it's very easy to predict which factors will lead to maximum probability of success in both outcome and landing.

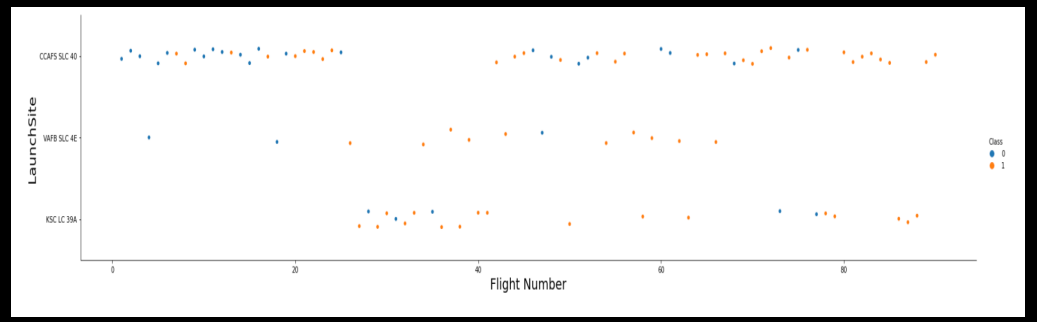


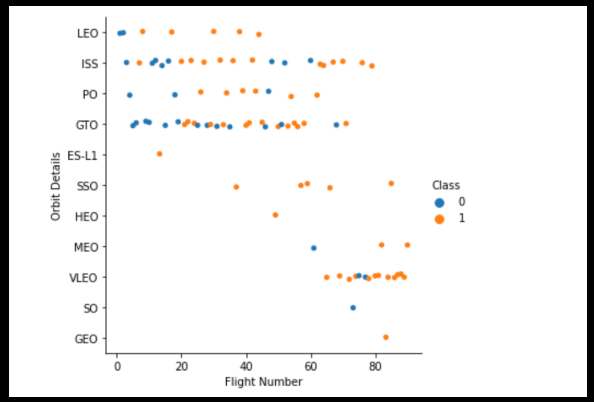
Dhaval Vibhakar

Payload vs. Launch Site

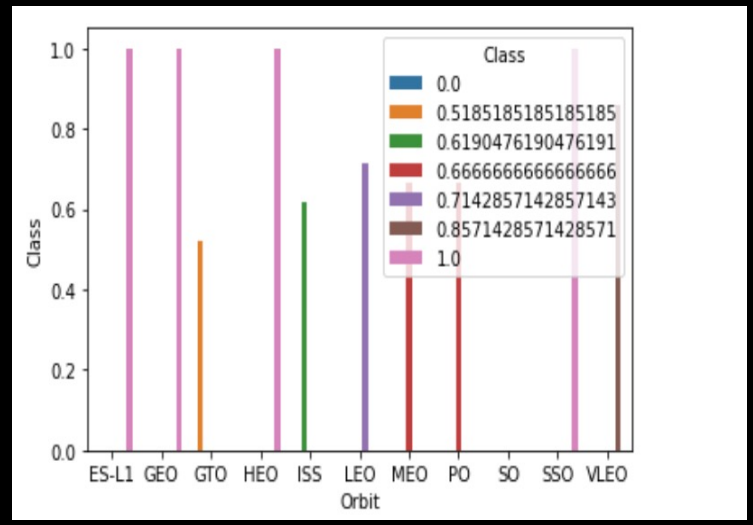


Flight Number vs. Launch Site

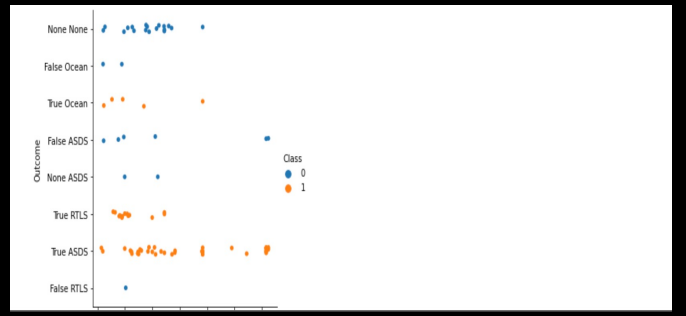




Flight Number vs. Orbit Type



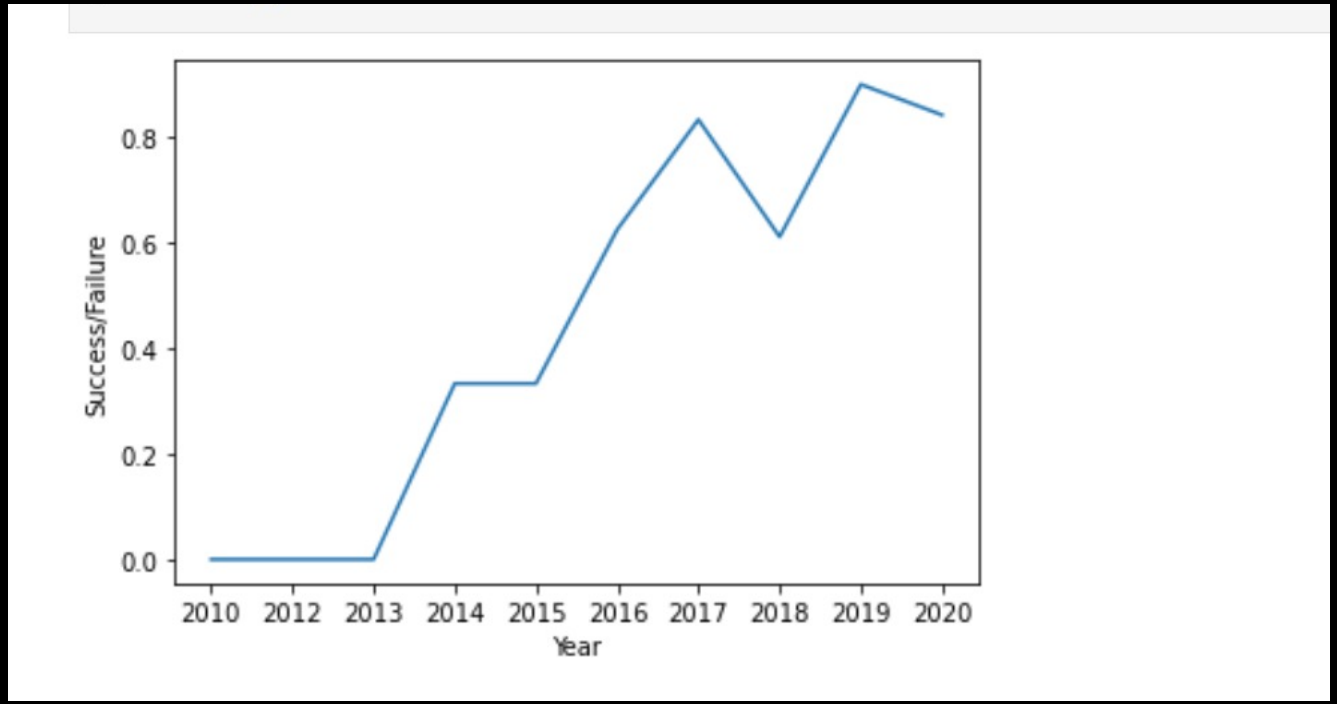
Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

Dhaval Vibhakar

Success/Failure vs Year



EDA with SQL

- SOL is an indispensable tool for Data Scientists and analysts as most of the real-world data is stored in databases. It's not only the standard language for
- Relational Database operations, but also an incredibly powerful tool for analyzing data and drawing useful insights from IBM's Db2 for Cloud, which is a fully managed SQL Database provided as a service.
- In this project we have used MySQL.
- we performed S&I Queries ot gather information from given datasét
- Displaying the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string C' CA'
- Displaying the total payload mass carried by boosters launched byNASA(CRS)
- Displaying average payload mass carried by booster version F9v1.1
- Listing the date where the successful landing outcome in drone ship was achieved
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000 Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass
- Listing the failed landingoutcomes in drone ship, their booster versions, and launch site names for the year 2015
- Ranking the count of landing outcomes (such as Failure (droneship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
import mysql.connector
import pandas as pd

# connect to MySQL database
cnx = mysql.connector.connect(user='root', password='Vib@131199',
                              host='127.0.0.1',
                              database='Coursera_sql')

# define the SQL query
query = """
SELECT *
FROM spacex
"""

# create a Pandas dataframe from the results of the query
df = pd.read_sql_query(query, cnx)

# print the dataframe

df
```

	Date	Time_UTC	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
...	...	...	...	...	...	...	...	...	...	...
96	2020-11-05	23:24:23	F9 B5B1062.1	CCAFS SLC-40	GPS III-04 , Crew-1	4311	MEO	USSF	Success	Success
97	2020-11-16	0:27:00	F9 B5B1061.1	KSC LC-39A	Crew-1, Sentinel-6 Michael Freilich	12500	LEO (ISS)	NASA (CCP)	Success	Success
98	2020-11-21	17:17:08	F9 B5B1063.1	VAFB SLC-4E	Sentinel-6 Michael Freilich, Starlink 15 v1.0	1192	LEO	NASA / NOAA / ESA / EUMETSAT	Success	Success
99	2020-11-25	2:13:00	F9 B5 B1049.7	CCAFS SLC-40	Starlink 15 v1.0, SpaceX CRS-21	15600	LEO	SpaceX	Success	Success
100	2020-12-06	16:17:08	F9 B5 B1058.4	KSC LC-39A	SpaceX CRS-21	2972	LEO (ISS)	NASA (CRS)	Success	Success

101 rows × 10 columns

Dhaval Vibhakar

The findings

- Unique launch sites in the space mission
- Launch Site Names begin with 'CCA’
- Total Payload Mass
- Average Payload Mass by F9 v1.1
- First Successful Ground Landing Date
- Successful Drone Ship Landing with Payload between 4000 and 6000
- Total number of Successful and Failure Mission Outcomes
- Boosters carried Maximum Payload
- 2015 Launch Records
- 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

Dhaval Vibhakar

Unique launch sites in the space mission

```
query1 = """
SELECT DISTINCT Launch_Site
FROM spacex;
"""

# create a Pandas dataframe from the results of the query
df1 = pd.read_sql_query(query1, cnx)
df1
```

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

- Launch Site Names begin with 'CCA’

```
In [83]: query2 = """
         SELECT LAUNCH_SITE from spacex where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
         """
         df2 = pd.read_sql_query(query2, cnx)
         df2
```

Out[83]:

	LAUNCH_SITE
0	CCAFS LC-40
1	CCAFS LC-40
2	CCAFS LC-40
3	CCAFS LC-40
4	CCAFS LC-40

Total Payload Mass

3	CCAFS LC-40
4	CCAFS LC-40

In [85]:
```
query3 = """
select sum(PAYLOAD_MASS_KG_) as payloadmass from spacex;
"""
df3 = pd.read_sql_query(query3, cnx)
df3
```

Out[85]:

	payloadmass
0	619967.0

Average Payload Mass by F9 v1.1

In [87]:
```
query4 = """
select avg(PAYLOAD_MASS_KG_) as payloadmass from spacex;
"""
df4 = pd.read_sql_query(query4, cnx)
df4
```

Out[87]:

	payloadmass
0	6138.2871

Dhaval Vibhakar

First Successful Ground Landing Date

In [89]:

```
query5 = """
select min(DATE) from spacex;
"""
df5 = pd.read_sql_query(query5, cnx)
df5
```

Out[89]:

	min(DATE)
0	2010-06-04

In [96]:
```
query6 = """
SELECT DISTINCT Booster_Version
FROM Spacex
WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS_KG_ BETWEEN 4000 AND 6000;
"""
df6 = pd.read_sql_query(query6, cnx)
df6
```

Out[96]:

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

Successful Drone Ship Landing with Payload between 4000 and 6000

In [97]:

```
query6 = """
select count(MISSION_OUTCOME) as missionoutcomes from spacex GROUP BY MISSION_OUTCOME;
"""
df6 = pd.read_sql_query(query6, cnx)
df6
```

Out[97]:

	missionoutcomes
0	99
1	1
2	1

Total number of Successful and Failure Mission Outcomes

Boosters carried Maximum Payload

In [100]:
```
query7 = """
SELECT Booster_Version, Payload_Mass_KG_
        FROM spacex
        WHERE Payload_Mass_KG_ = (
                                  SELECT MAX(Payload_Mass_KG_)
                                  FROM spacex
                                  )
        ORDER BY Booster_Version
"""
df7 = pd.read_sql_query(query7, cnx)
df7
```

Out[100]:

	Booster_Version	Payload_Mass_KG_
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

In [101]:
```
query8 = """
SELECT Booster_Version, Launch_Site, Landing_Outcome
        FROM spacex
        WHERE Landing_Outcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
"""
df8 = pd.read_sql_query(query8, cnx)
df8
```

Out[101]:

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

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

In [102]:

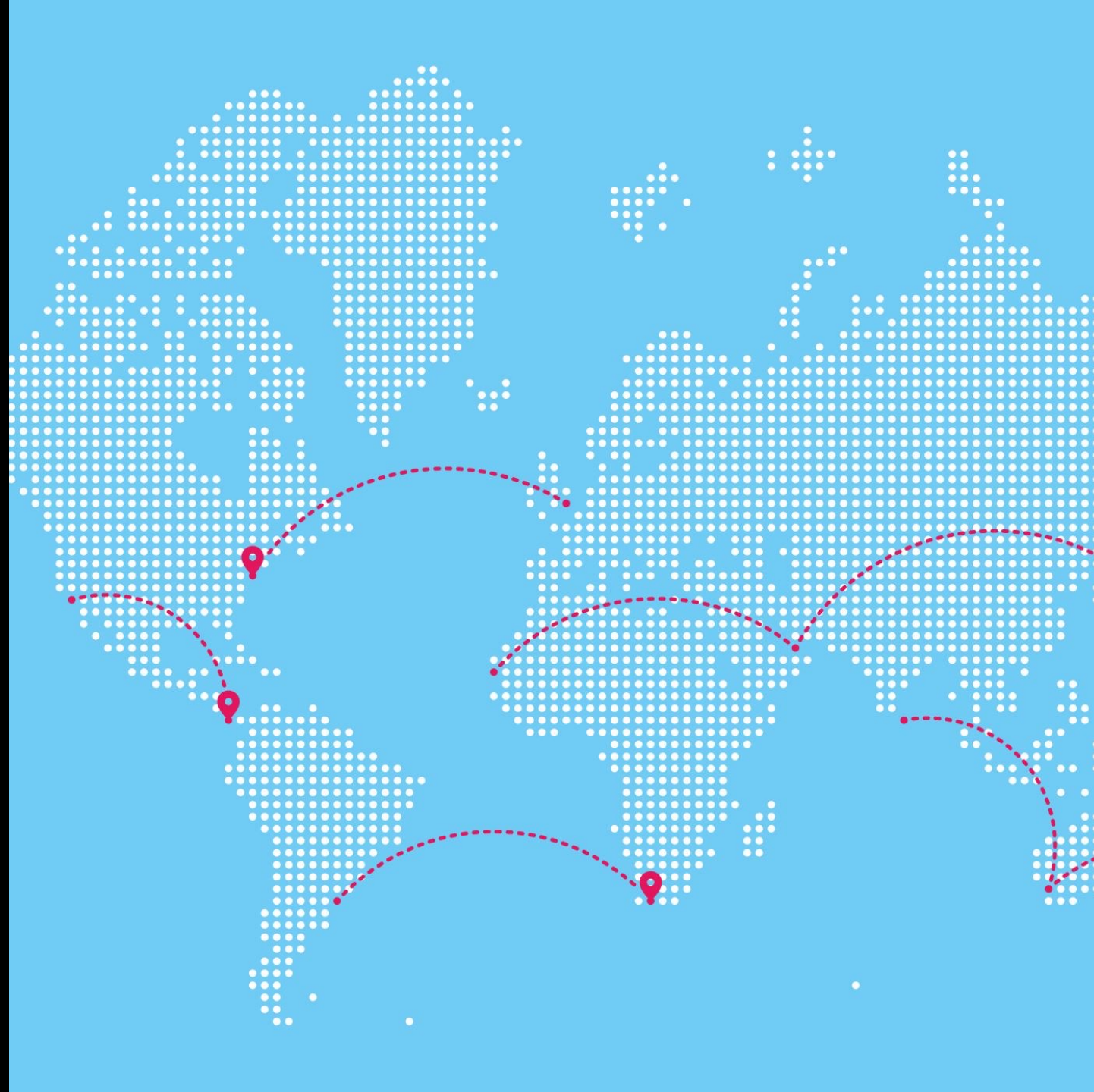
```
query9 = """
SELECT Landing_Outcome, COUNT(Landing_Outcome)
        FROM spacex
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY Landing_Outcome
        ORDER BY COUNT(Landing_Outcome) DESC
"""
df9 = pd.read_sql_query(query9, cnx)
df9
```

Out[102]:

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

Build an Interactive Map with Folium

- Folium makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map. We use the latitude and longitude coordinates for each launch site and added a Circle Marker around each launch site with a label of the name of the launch site. It is also easy to visualize the number of success and failure for each launch site with Green and Red markers on the map.



Dhaval Vibhakar

All Launch Sites on Folium Map

In [4]:
```
# 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
```

Out[4]:

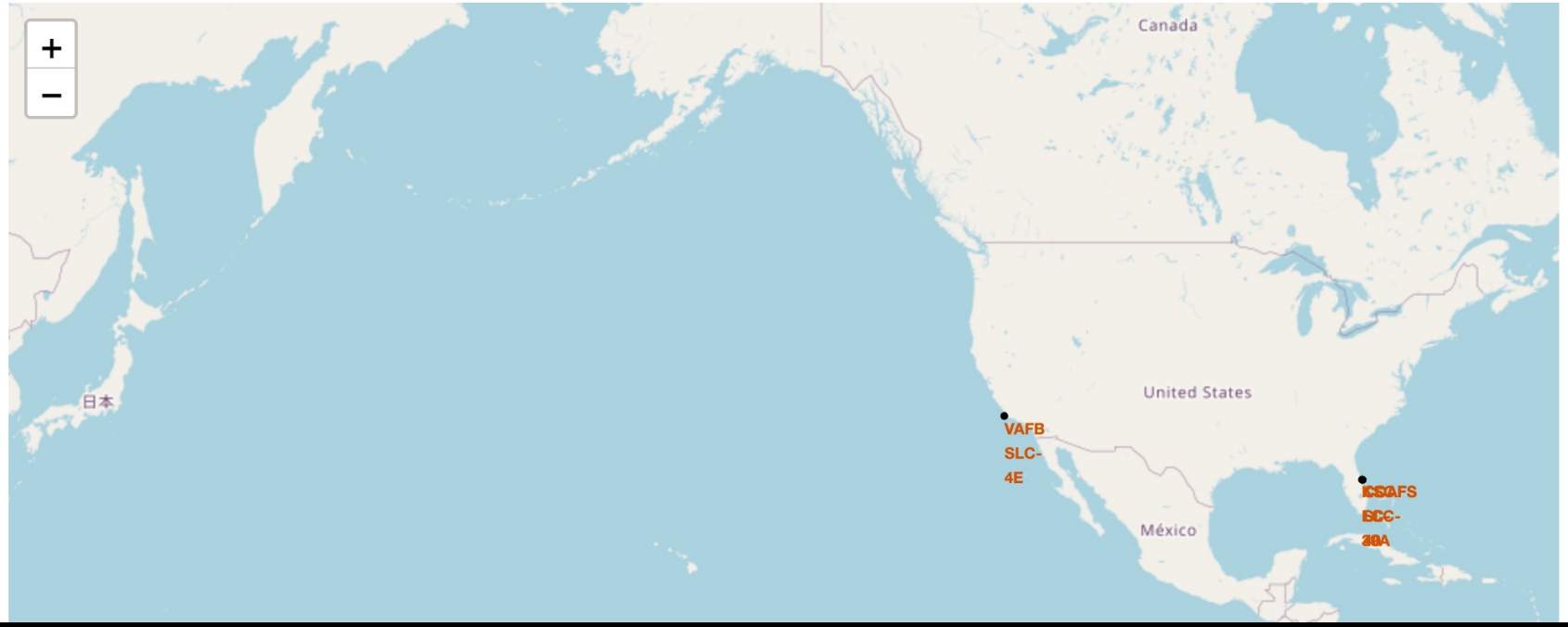
	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Dhaval Vibhakar

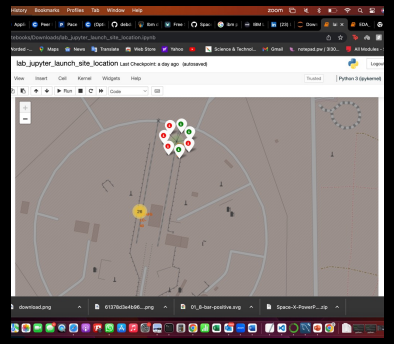
All Launch Sites on Folium Map

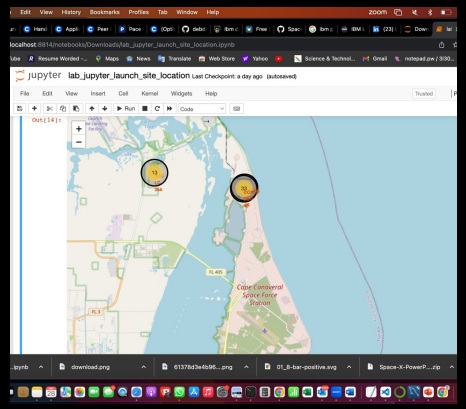
In [7]:
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site na
for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(row['Launch Site'])).add_
    folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; cc
site_map
```

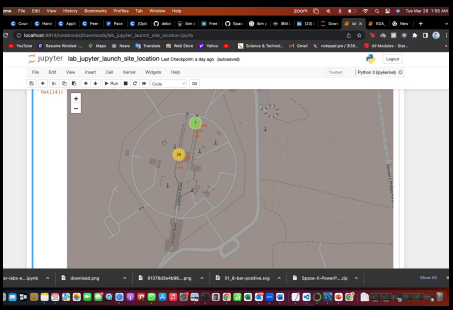
Out[7]:

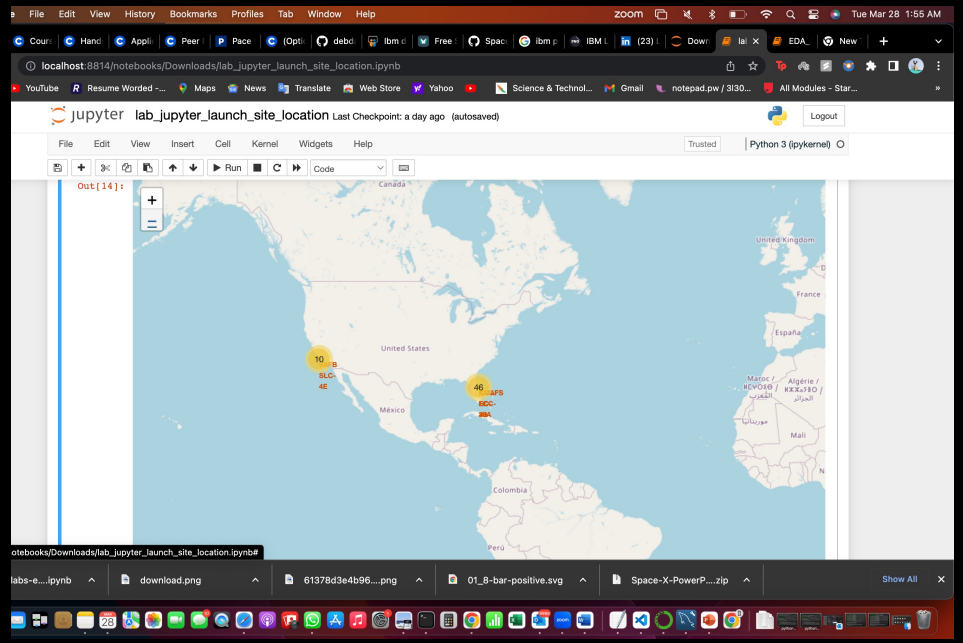


Color Labeled Launch Records









Dhaval Vibhakar

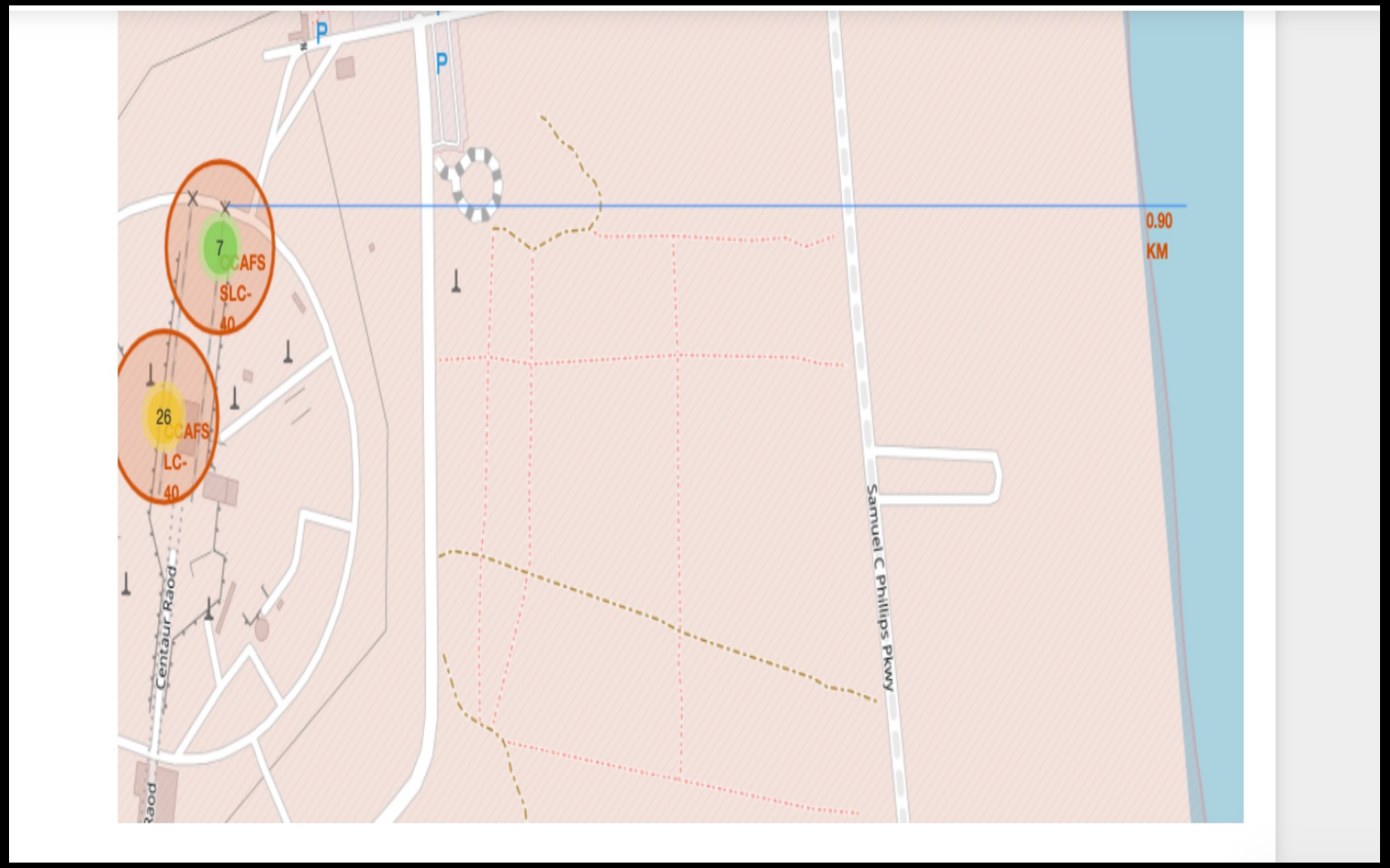
Launch Site Distances from Equator &Railways

TODO: Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

In [17]:
```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
launch_site_lat = 28.563197
launch_site_lon = -80.576820
coastline_lat = 28.56334
coastline_lon = -80.56799
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
print(distance_coastline,' km')
```

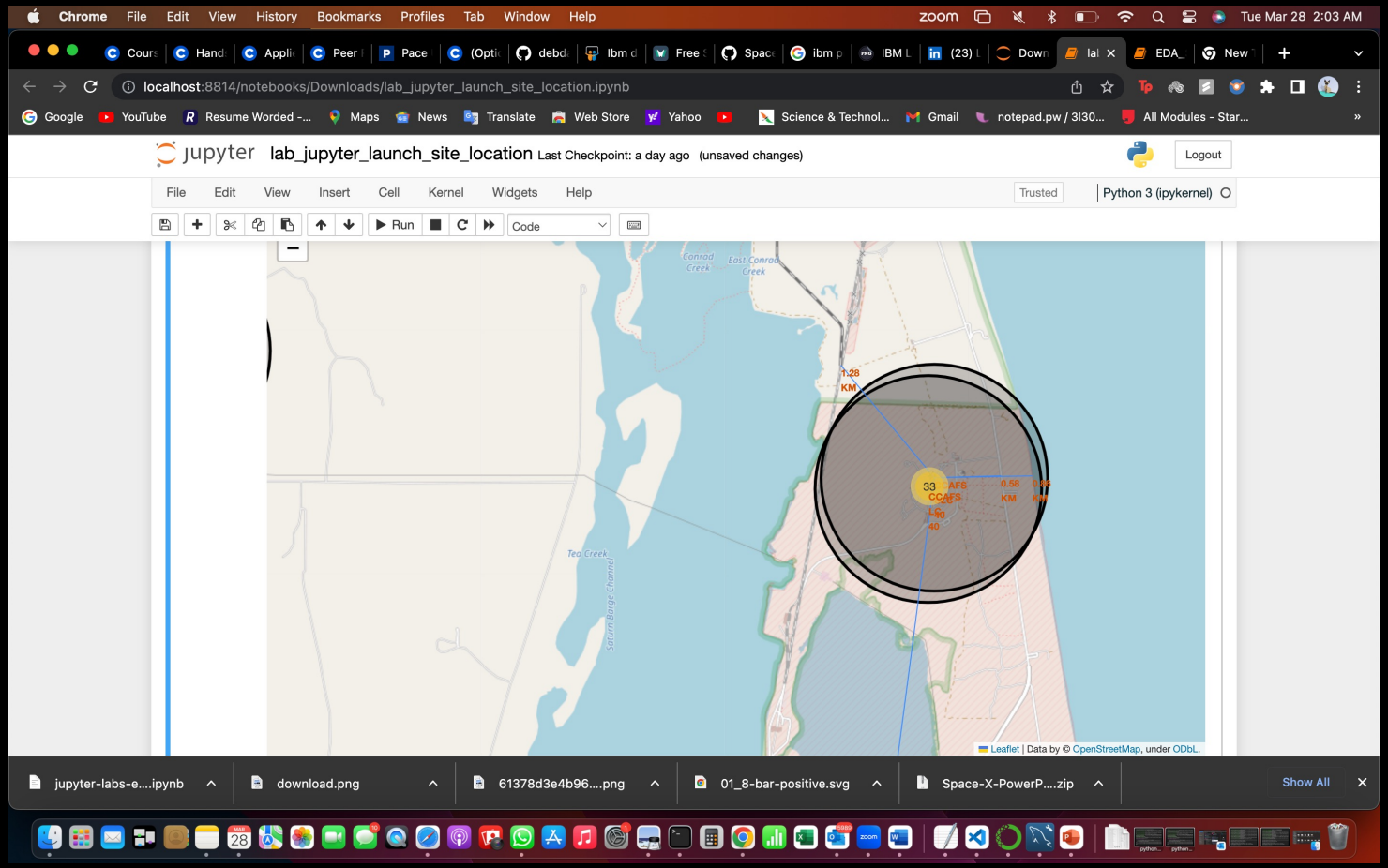
```
0.8627671182499878  km
```

Launch Site Distances from Coastlines & Cities

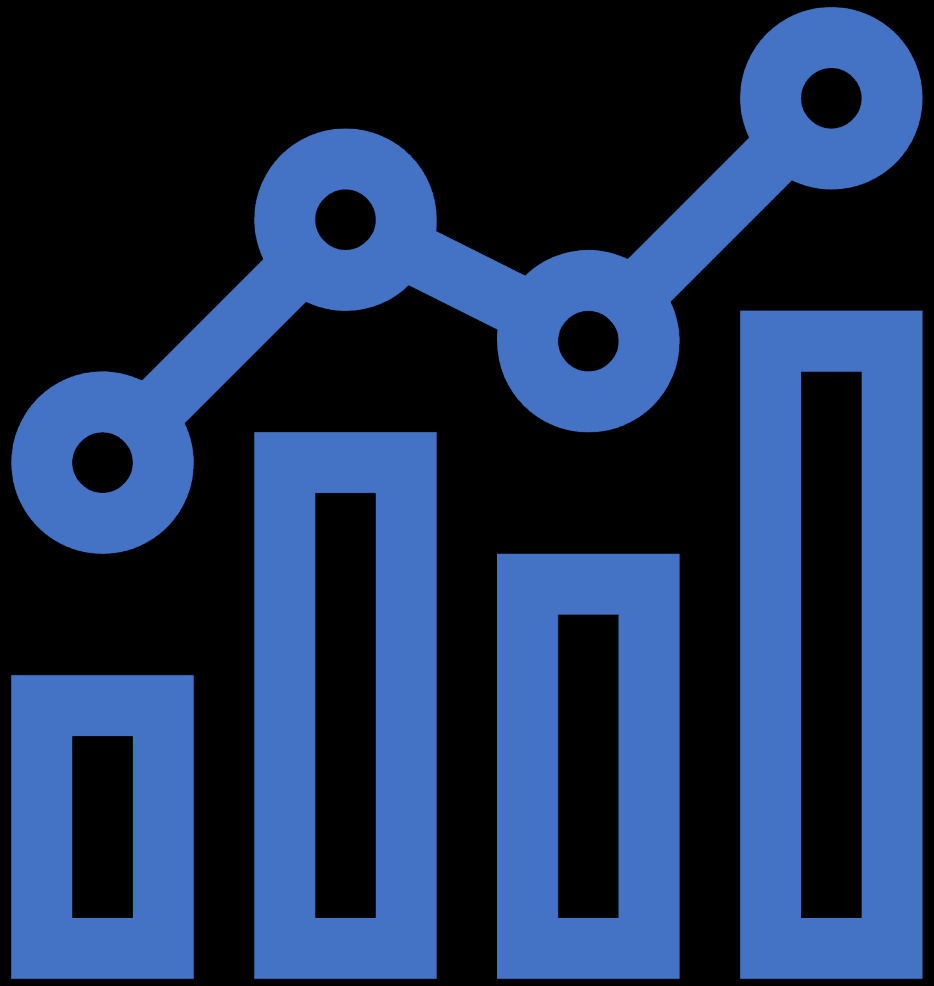


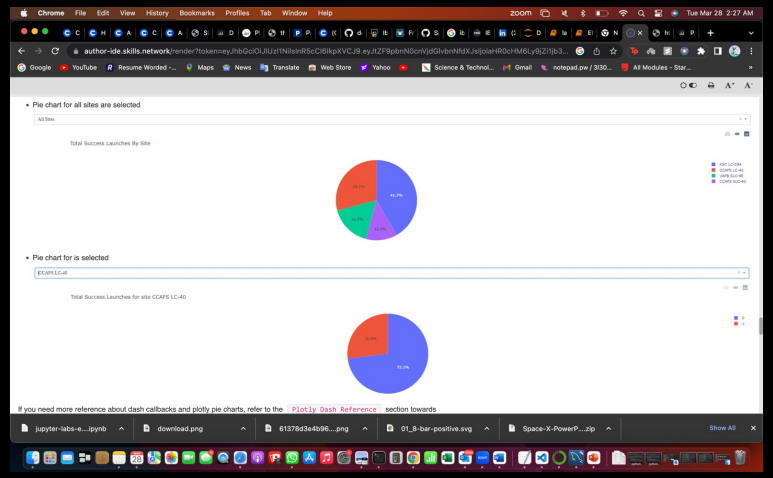
Dhaval Vibhakar

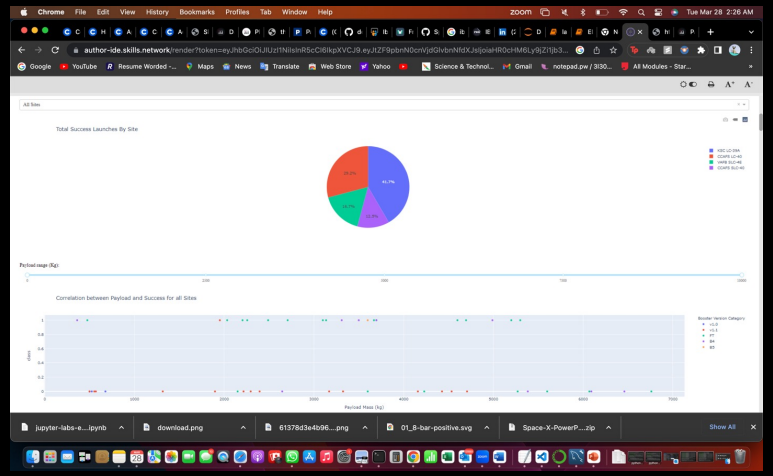
Launch Site Distances from Highways

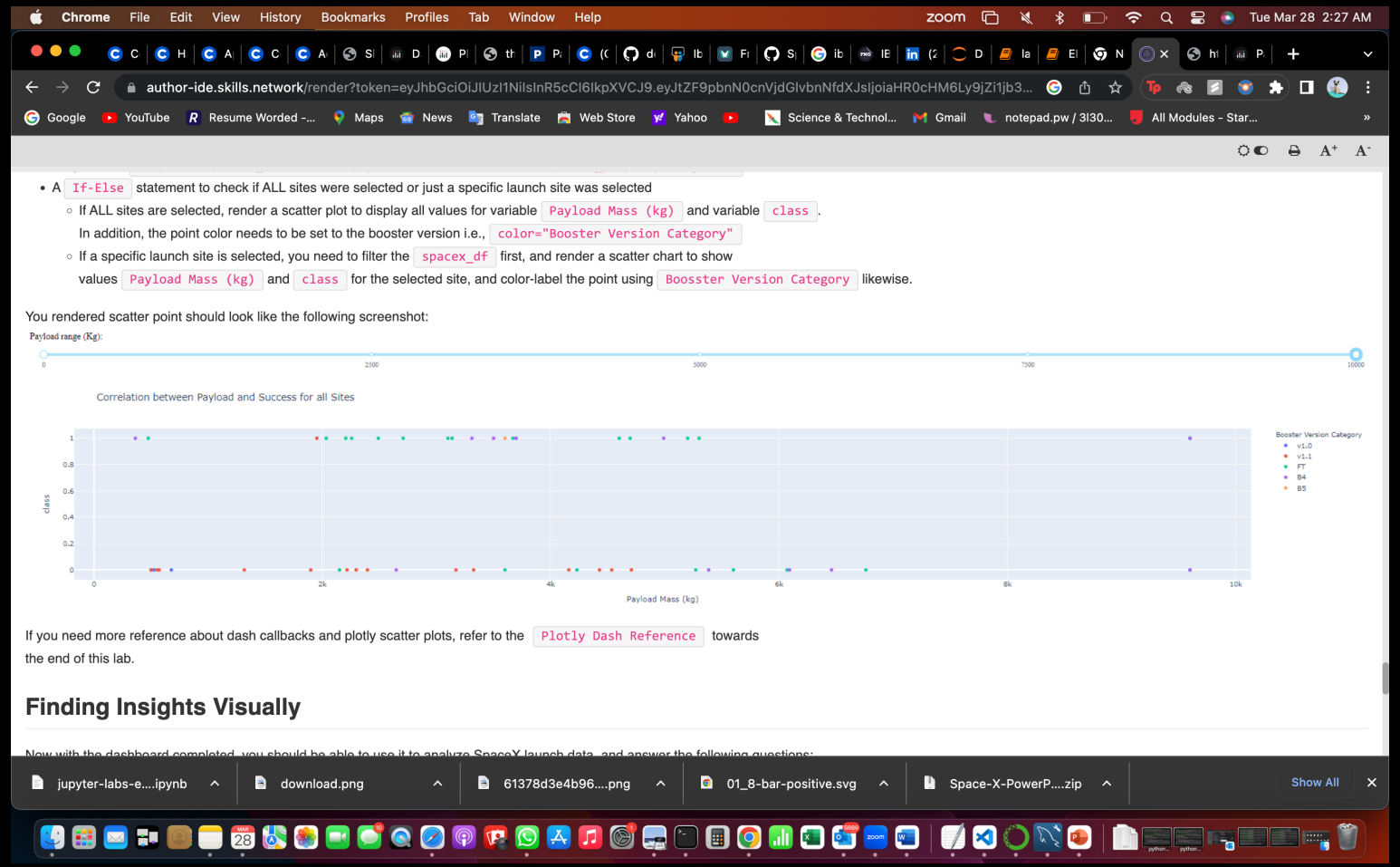


Analyze with Ploty Dashboard





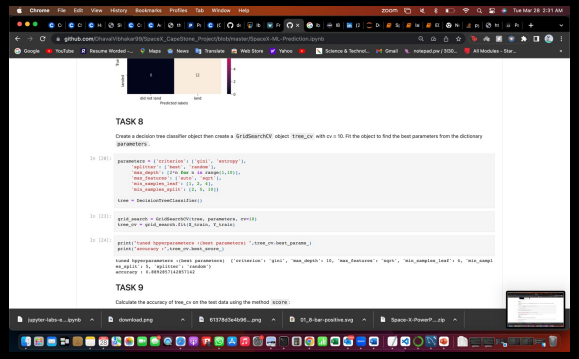




Plotly Dashboard

Dhaval Vibhakar

Predictive analysis



TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[10]:
parameters ={'C':[0.01,0.1,1],
              'penalty':['l2'],
              'solver':['lbfgs']}

lr=LogisticRegression()
grid_search = GridSearchCV(lr, parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)
```

```
[11]:
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
[12]:
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

```
In [8]:
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

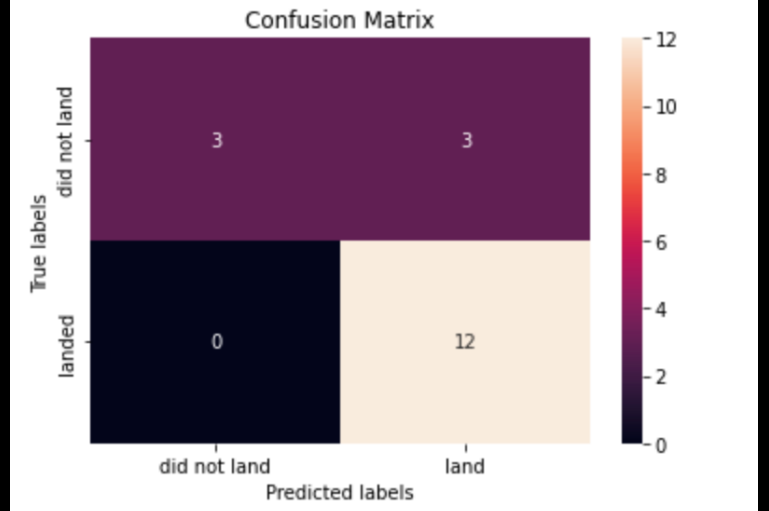
```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

```
In [9]:
Y_test.shape
```

```
Out[9]: (18,)
```

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



TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

```
[21]:
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

```
[22]:
grid_search = GridSearchCV(KNN, parameters, cv=10)
knn_cv = grid_search.fit(X_train, Y_train)
```

```
[23]:
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

```
tree_cv.score(X_test, Y_test)
```

```
0.6111111111111112
```

We can plot the confusion matrix

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

```
In [5]:
Y = data['Class'].to_numpy()
Y
```

```
Out[5]: array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
               1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
               1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
               1, 1])
```

Conclusion

- Orbits ES-L1, GEO, HEO, SSO has highest Sucess rates
- Success rates for Space 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 inceasing payload mass seems to have negative impact on success
- Decision Tree Classifier Algorithm is the best for Machine Learning Model for provided dataset

Thank you