Winning Space Race with Data Science

IBM Data Science Professional Certificate
- Applied Data Science Capstone Project -
- Final Documentation -

Dominik Hahn
December 26, 2022

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection (Collecting data from the SpaceX API and from Wikipedia using Webscraping)
  - Data wrangling
  - Exploratory Data Analytics with Data Visualization
  - Exploratory Data Analytics with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- **Summary of all results**
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

Introduction

- **Project background and context**
  - 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.
  - 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.
  - This presentation documents the approach, process and results to predict if the Falcon 9 first stage will land successfully.
- **Problems you want to find answers**
  - Correlations between each rocket variable and successful landing rate
  - Conditions to get the best results and ensure the best successful landing rate

Section 1

Methodology

Methodology (1/2)

Executive Summary

- Data collection methodology:
  - We applied to methods in order to collect the relevant data:
    1) Collecting data from the SpaceX API
    2) Collecting data from Wikipedia using Webscraping
- Perform data wrangling:
  - We added outcome labels to the dataset

Methodology (2/2)

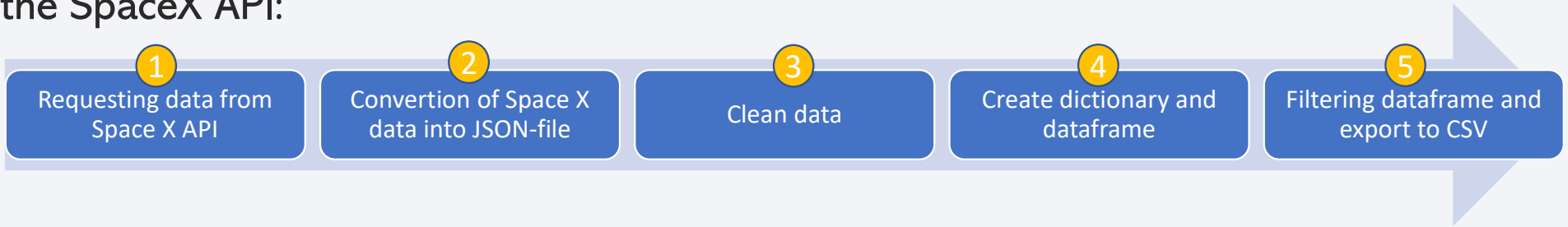
Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using Scatter, Bar and Line Charts to visualize the data
  - Using SQL programming language to perform queries and present results
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Folium to locate and show the launch sites on the world map
  - Using Plotly Dash to create charts to evaluate the data
- Perform predictive analysis using classification models
  - We used the models Logistic Regression, Support Vector Machine, Decision Tree Classifier and K nearest neighbors in order to analyze the data

Data Collection (from SpaceX API)

1) Data Collection with API request

The following flowchart presents the data collection process using API requests from the SpaceX API:



In detail, the API request was performed as follows:

1. Requesting rocket launch data from SpaceX API with the following URL:

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

response = requests.get(spacex_url)
```

Data Collection (from SpaceX API)

Follow-up:

2. Decoding the response content as a Json string and turn it into a Pandas dataframe:

```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

3. Clean data:

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

Data Collection (from SpaceX API)

Follow-up:

3. Clean data:

```
# Call getPayloadData
getPayloadData(data)
```

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

```
# Call getCoreData
getCoreData(data)
```

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Data Collection (from SpaceX API)

Follow-up:

4. Create dicionary and dataframe:

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

```
# Show the head of the dataframe
df.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

Data Collection (from SpaceX API)

Follow-up:

5. Filtering data and export to CSV:

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df["BoosterVersion"] == 'Falcon 9']
```

Replacing missing values:

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9["PayloadMass"].mean()
print("Average of PayloadMass", mean)
```

```
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, mean, inplace=True)
```

```
data_falcon9.head(1000)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857
5	2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366	28.561857
6	3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366	28.561857
7	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829	34.632093
8	5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

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

Data Collection (from SpaceX API)

Github-URL:

Final Data Science Capstone Project - Week 1 - Lab 1 - SpaceX Data Collection API.ipynb

Data Collection (from Wikipedia)

2) Data Collection with Webscraping

The following flowchart presents the data collection process using Webscraping:

In detail, the API request was performed as follows:

1. Requesting Falcon9 Launch HTML page using HTTP GET Method:

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url).text
```

Data Collection (from Wikipedia)

Follow-up:

2020 [edit]

In late 2019, Gwynne Shotwell stated that SpaceX hoped for as many as 24 launches for Starlink satellites in 2020,[490] in addition to 14 or 15 non-Starlink launches. At 26 launches, 13 of which for Starlink satellites, Falcon 9 had its most prolific year, and Falcon rockets were second most prolific rocket family of 2020, only behind China's Long March rocket family.[491]

[hide] Flight No.	Date and time (UTC)	Version, Booster[b]	Launch site	Payload[c]	Payload mass	Orbit	Customer	Launch outcome	Booster landing
78	7 January 2020, 02:19:21[492]	F9 B5 △ B1049.4	CCAFS, SLC-40	Starlink 2 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third large batch and second operational flight of Starlink constellation. One of the 60 satellites included a test coating to make the satellite less reflective, and thus less likely to interfere with ground-based astronomical observations.[493]								
79	19 January 2020, 15:30[494]	F9 B5 △ B1046.4	KSC, LC-39A	Crew Dragon in-flight abort test[495] (Dragon C205.1)	12,050 kg (26,570 lb)	Sub-orbital[496]	NASA (CTS)[497]	Success	No attempt
	An atmospheric test of the Dragon 2 abort system after Max Q. The capsule fired its SuperDraco engines, reached an apogee of 40 km (25 mi), deployed parachutes after reentry, and splashed down in the ocean 31 km (19 mi) downrange from the launch site. The test was previously slated to be accomplished with the Crew Dragon Demo-1 capsule,[498] but that test article exploded during a ground test of SuperDraco engines on 20 April 2019.[419] The abort test used the capsule originally intended for the first crewed flight.[499] As expected, the booster was destroyed by aerodynamic forces after the capsule aborted.[500] First flight of a Falcon 9 with only one functional stage — the second stage had a mass simulator in place of its engine.								
80	29 January 2020, 14:07[501]	F9 B5 △ B1051.3	CCAFS, SLC-40	Starlink 3 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)
	Third operational and fourth large batch of Starlink satellites, deployed in a circular 290 km (180 mi) orbit. One of the fairing halves was caught, while the other was fished out of the ocean.[502]								
81	17 February 2020, 15:05[503]	F9 B5 △ B1056.4	CCAFS, SLC-40	Starlink 4 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fourth operational and fifth large batch of Starlink satellites. Used a new flight profile which deployed into a 212 km × 386 km (132 mi × 240 mi) elliptical orbit instead of launching into a circular orbit and firing the second stage engine twice. The first stage booster failed to land on the drone ship[504] due to incorrect wind data.[505] This was the first time a flight proven booster failed to land.								
82	7 March 2020, 04:50[506]	F9 B5 △ B1059.2	CCAFS, SLC-40	SpaceX CRS-20 (Dragon C112.3 △)	1,977 kg (4,359 lb)[507]	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
	Last launch of phase 1 of the CRS contract. Carries *Bartolomeo*, an ESA platform for hosting external payloads onto ISS.[508] Originally scheduled to launch on 2 March 2020, the launch date was pushed back due to a second stage engine failure. SpaceX decided to swap out the second stage instead of replacing the faulty part.[509] It was SpaceX's 50th successful landing of a first stage booster, the third flight of the Dragon C112 and the last launch of the cargo Dragon spacecraft.								
83	18 March 2020, 12:16[510]	F9 B5 △ B1048.5	KSC, LC-39A	Starlink 5 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Failure (drone ship)
	Fifth operational launch of Starlink satellites. It was the first time a first stage booster flew for a fifth time and the second time the fairings were reused (Starlink flight in May 2019).[511] Towards the end of the first stage burn, the booster suffered premature shut down of an engine, the first of a Merlin 1D variant and first since the CRS-1 mission in October 2012. However, the payload still reached the targeted orbit.[512] This was the second Starlink launch booster landing failure in a row, later revealed to be caused by residual cleaning fluid trapped inside a sensor.[513]								
84	22 April 2020, 19:30[514]	F9 B5 △ B1051.4	KSC, LC-39A	Starlink 6 v1.0 (60 satellites)	15,600 kg (34,400 lb)[5]	LEO	SpaceX	Success	Success (drone ship)

Data Collection (from Wikipedia)

Follow-up:

2. Creating a Beautiful Soup object:

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(page, "html.parser")
```

3. Find table No. 3 on HTML-Page:

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
```

```
# Let's print the third table and check its content
first_launch_table = (html_tables)[2]
print(first_launch_table)
```

Data Collection (from Wikipedia)

Follow-up:

4. Extract column names from table:

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

Data Collection (from Wikipedia)

Follow-up:

5. Create dataframe and export to CSV:

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Collection (from Wikipedia)

Follow-up:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number) #TODO-1
            #print(flight_number)
            datatimelist=date_time(row[0])
---------------
            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date) #TODO-2
            #print(date)
---------------
            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time) #TODO-3
            #print(time)
----------------
            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv) #TODO-4
            #print(bv)
```

```
---------------
            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site) #TODO-5
            #print(launch_site)
---------------
            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload) #TODO-6
            #print(payload)
---------------
            # Payload Mass
            # TODO: Append the payload_mass into launch_dict with key `Payload mass`
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass) #TODO-7
            #print(payload)
---------------
            # Orbit
            # TODO: Append the orbit into launch_dict with key `Orbit`
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit) #TODO-8
            #print(orbit)
---------------
            # Customer
            # TODO: Append the customer into launch_dict with key `Customer`
            customer = row[6].text.strip()
            launch_dict['Customer'].append(customer) #TODO-9
            #print(customer)
---------------
            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome) #TODO-10
            #print(launch_outcome)
---------------
            # Booster Landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster Landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing) #TODO-11
            #print(booster_landing)
---------------
```

Data Collection (from Wikipedia)

Follow-up:

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

	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 (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX Capella Space and Tyvak	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA (CRS)	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

121 rows × 11 columns

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

Data Collection (from Wikipedia)

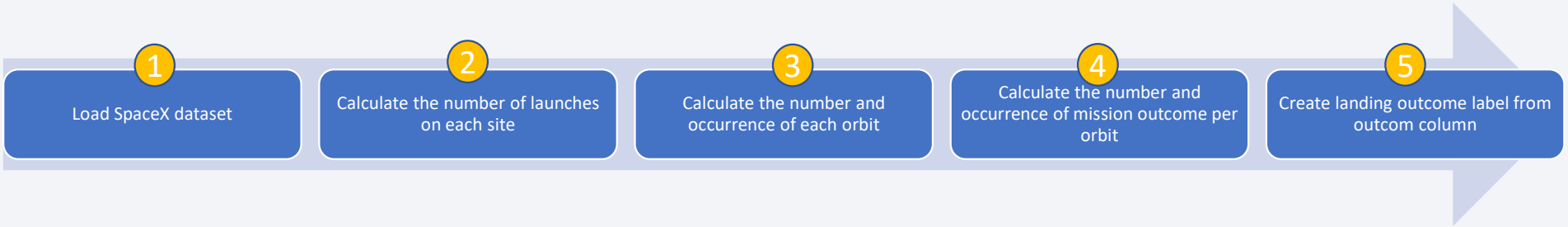
Github-URL:

Final Data Science Capstone Project - Week 1 - Lab 2 - SpaceX Webscraping.ipynb

Data Wrangling

3) Data Wranging Process

The following flowchart presents the data wrangling process:



In detail, the Data Wrangling process was performed as follows:

1. Load SpaceX dataset:

```
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)
```

Data Wrangling

Follow-up:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

Data Wrangling

Follow-up:

2. Calculate the number of launches on each site:

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

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

3. Calculate the number and occurrence of each orbit:

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

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

Data Wrangling

Follow-up:

4. Calculate the number and occurrence of mission outcome per orbit type:

```
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts('Outcome')
print(landing_outcomes)
```

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
dtype: int64
```

Data Wrangling

Follow-up:

5. Create a landing outcome label from Outcome column:

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

def assignNewLabel(label):
    if label == 'True ASDS' or label == 'True RTLS' or label == 'True Ocean':
        return 1
    else:
        return 0

landing_class = df['Outcome'].apply(assignNewLabel)
```

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

Data Wrangling

Follow-up:

5 Create a landing outcome label from Outcome column:

```
df.head(20)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857	0
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857	1
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857	1
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857	0
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857	0
10	11	2014-09-21	Falcon 9	2216.000000	ISS	CCAFS SLC 40	False Ocean	1	False	False	False	NaN	1.0	0	B1010	-80.577366	28.561857	0
11	12	2015-01-10	Falcon 9	2395.000000	ISS	CCAFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb761634e7cb	1.0	0	B1012	-80.577366	28.561857	0
12	13	2015-02-11	Falcon 9	570.000000	ES-L1	CCAFS SLC 40	True Ocean	1	True	False	True	NaN	1.0	0	B1013	-80.577366	28.561857	1
13	14	2015-04-14	Falcon 9	1898.000000	ISS	CCAFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb761634e7cb	1.0	0	B1015	-80.577366	28.561857	0
14	15	2015-04-27	Falcon 9	4707.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1016	-80.577366	28.561857	0
15	16	2015-06-28	Falcon 9	2477.000000	ISS	CCAFS SLC 40	None ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	1.0	0	B1018	-80.577366	28.561857	0
16	17	2015-12-22	Falcon 9	2034.000000	LEO	CCAFS SLC 40	True RTLS	1	True	False	True	5e9e3032383ecb267a34e7c7	1.0	0	B1019	-80.577366	28.561857	1
17	18	2016-01-17	Falcon 9	553.000000	PO	VAFB SLC 4E	False ASDS	1	True	False	True	5e9e3033383ecbb9e534e7cc	1.0	0	B1017	-120.610829	34.632093	0
18	19	2016-03-04	Falcon 9	5271.000000	GTO	CCAFS SLC 40	False ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	1.0	0	B1020	-80.577366	28.561857	0
19	20	2016-04-08	Falcon 9	3136.000000	ISS	CCAFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	2.0	1	B1021	-80.577366	28.561857	1

Data Wrangling

Github-URL:

Final Data Science Capstone Project - Week 1 - Lab 3 - SpaceX Data Wrangling.ipynb

Exploratory Data Analysis with Data Visualization

- We created the following charts to solve the assignment tasks:

1) Scatter Charts:

- Task 1: Flight Number vs. Orbit Type
- Task 2: Flight Number vs. Launch Site
- Task 3: Payload Mass vs. Launch Site
- Task 4: Flight Number vs. Orbit Type
- Task 5: Payload Mass vs. Orbit Type

A scatter plot shows how much one variable is affected by another. The relationship between two variables is called a correlation. This plot is generally composed of large data bodies.

Exploratory Data Analysis with Data Visualization

- We created the following charts to solve the assignment tasks:

2) Bar Charts:

- Task 6: Orbit Type vs. Success Rate

A Bar chart makes it easy to compare datasets between multiple groups at a glance. One axis represents a category and the other axis represents a discrete value. The purpose of this chart is to indicate the relationship between the two axes.

3) Line Charts:

- Task 7: Year vs. Success Rate

A Line chart shows data variables and trends very clearly and helps predict the results of data that has not yet been recorded.

Exploratory Data Analysis with Data Visualization

Github-URL:

Final Data Science Capstone Project - Week 2 - Lab 2 - SpaceX EDA with Data Visualization.ipynb

Exploratory Data Analysis with SQL

- We loaded the SQL extension and establish a connection with the database:

```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

```
import csv, sqlite3

con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

```
!pip install -q pandas==1.1.5
```

```
%sql sqlite:///my_data1.db
```

```
'Connected: @my_data1.db'
```

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv")
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

Exploratory Data Analysis with SQL

- We performed the following SQL queries to solve the assignment tasks:
  - Task 1: Display the names of the unique launch sites in the space mission
  - Task 2: Display 5 records where launch sites begin with the string 'CCA'
  - Task 3: Display the total payload mass carried by boosters launched by NASA (CRS)
  - Task 4: Display average payload mass carried by booster version F9 v1.1
  - Task 5: List the date when the first successful landing outcome in ground pad was achieved
  - Task 6: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Task 7: List the total number of successful and failure mission outcomes

Exploratory Data Analysis with SQL

- Follow-up:
  - Task 8: List the names of the booster versions which have carried the maximum payload mass. Use a subquery
  - Task 9: 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
  - Task 10: Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Exploratory Data Analysis with SQL

Github-URL:

Final Data Science Capstone Project - Week 2 - Lab 1 - SpaceX EDA with SQL.ipynb

Build an Interactive Map with Folium

- Objects created and added to a folium map:
  - Markers that show all launch sites on a map
  - Markers that show the success/failed launches for each site on the map
  - Lines that show the distances between a launch site to its proximities
- By adding these objects, following geographical patterns about launch sites are found:
  - Are launch sites in close proximity to railways? Yes
  - Are launch sites in close proximity to highways? Yes
  - Are launch sites in close proximity to coastline? Yes
  - Do launch sites keep certain distance away from cities? Yes

Build an Interactive Map with Folium

Github-URL:

Final Data Science Capstone Project - Week 3 - Lab 1 - Interactive Visual Analytics.ipynb

Build a Dashboard with Plotly Dash

- We created a dashboard application containing pie charts and scatter charts:

1) Pie Charts:

- For showing total success launches by sites
- This chart can be selected to indicate a successful landing distribution across all launch

2) Scatter Charts:

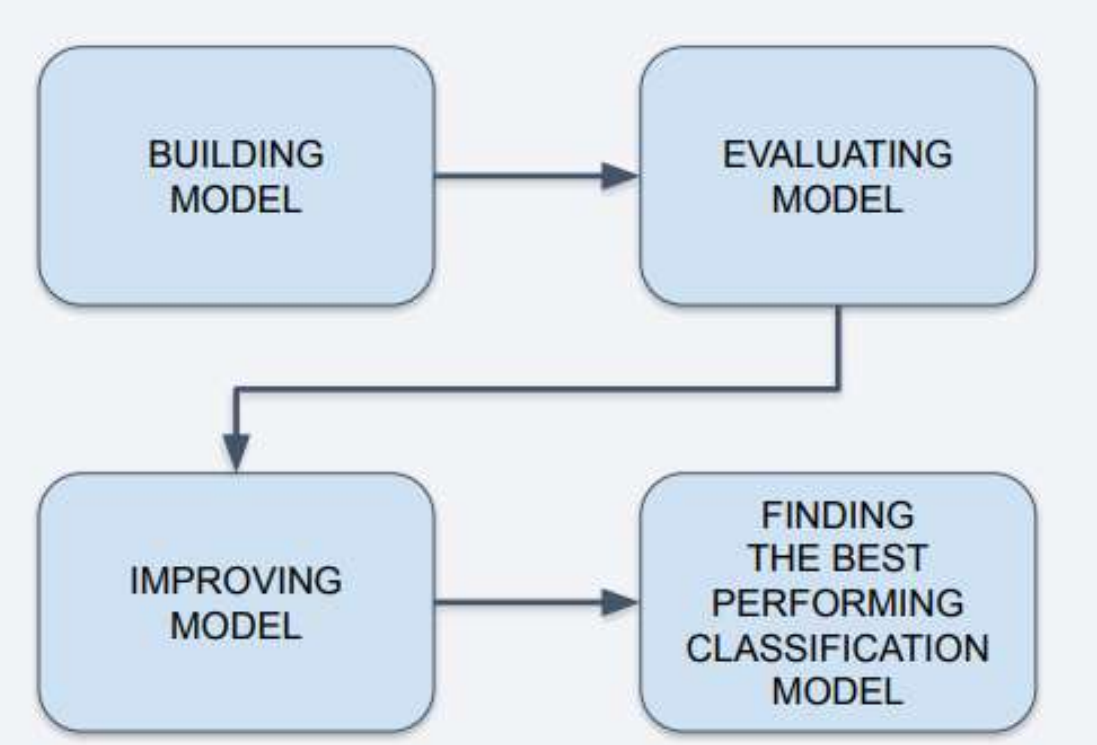
- For showing the relationship between Outcomes and Payload mass(Kg) by different boosters
- Has 2 inputs: All sites/individual site & Payload mass on a slider between 0 and 10000 kg
- This chart helps determine how success depends on the launch point, payload mass, and booster version categories

Build a Dashboard with Plotly Dash

Github-URL:

Final Data Science Capstone Project - Week 3 - Lab 2 - Ploty Dash-App.py

Predictive Analysis (Classification)



- Perform exploratory Data Analysis and determine Training Labels
  - Create a column for the class
  - Standardize the data
  - Split into training data and test data
- Find the method performs best using test data
- We used the models Logistic Regression, Support Vector Machine, Decision Tree Classifier and K nearest neighbors

Predictive Analysis (Classification)

Github-URL:

Final Data Science Capstone Project - Week 4 - Lab 1 - Machine Learning Prediction.ipynb

Results

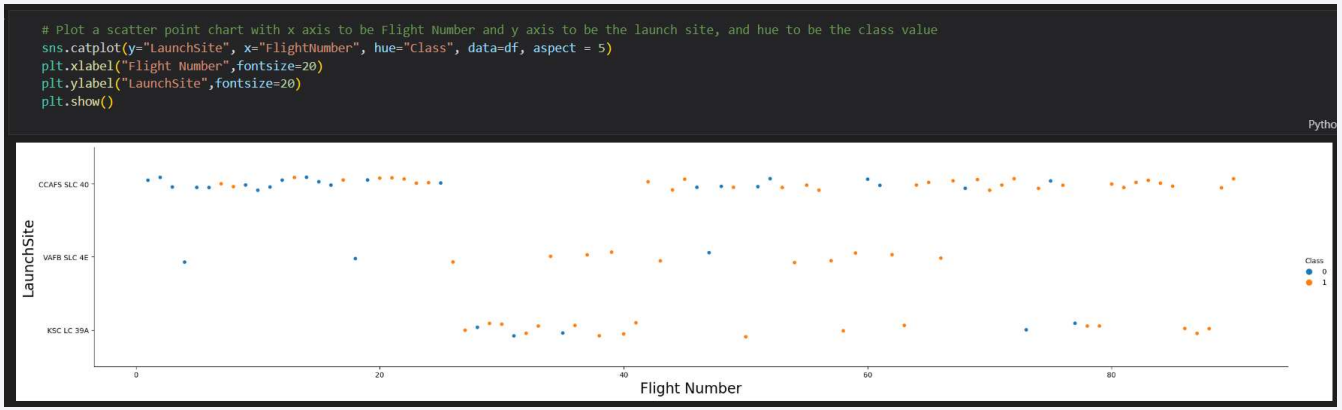
- Orbit types SSO, HEO, GEO and ES-L1 have the highest success rate (100%).
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites.
- As the number of flights increased, the success rate increased, and recently it has exceeded 80%.
- The launch site is close to railways, highways, and coastline, but far from cities.
- The launch success rate of low weighted payloads is higher than that of heavy weighted payloads.
- In this dataset, all models have the same accuracy (83.33%).


Section 2
Insights drawn from EDA
- Exploratory Data Analysis with Data Visualization -

EDA with Data Visualization - Task 1

1 Task 1: Flight Number vs. Launch Site (Scatter Chart)

```
# 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 = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Legend:

- Class 0 (blue) represents unsuccessful launches.
- Class 1 (orange) represents successful launches.

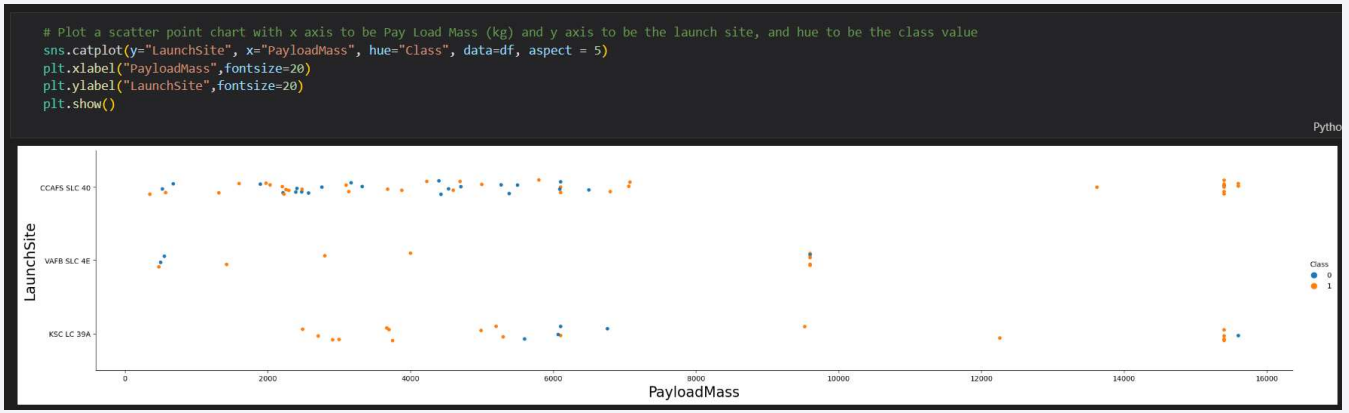
Explanations:

- Launch site CCAFS SLC 40 success rate increased with the increase in flights from this site.
- Launch site VAFB SLC 4E has only 3 unsuccessful launches.
- Launch site KSC LC 39A has 5 unsuccessful launches, 3 of them within the 8 first flights from this site.

EDA with Data Visualization - Task 2

② Task 2: Payload Mass vs. Launch Site (Scatter Chart)

```
# 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 = 5)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



Legend:

- Class 0 (blue) represents unsuccessful launches.
- Class 1 (orange) represents successful launches.

Explanations:

- Launch site CCAFS SLC 40 success rate increases with the increase in the payload mass.
- Launch site VAFB SLC 4E success rate increases with the increase in the payload mass.
- Launch site KSC LC 39A success rate increases with the increase in the payload mass.

EDA with Data Visualization - Task 3

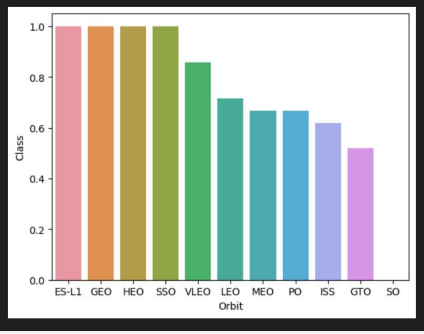
③ Task 3: Orbit Type vs. Success Rate (Bar Chart)

```
# HINT use groupby method on Orbit column and get the mean of Class column
data = df
success_data = data.groupby('Orbit')['Class'].mean().reset_index()
success_data
```

	Orbit	Class
0	ES-L1	1.000000
1	GEO	1.000000
2	GTO	0.518519
3	HEO	1.000000
4	ISS	0.619048
5	LEO	0.714286
6	MEO	0.666667
7	PO	0.666667
8	SO	0.000000
9	SSO	1.000000
10	VLEO	0.857143

```
sns.barplot(data=success_data, x="Orbit", y="Class", order=success_data.sort_values('Class',ascending = False).Orbit)
```

```
<AxesSubplot:xlabel='Orbit', ylabel='Class'>
```



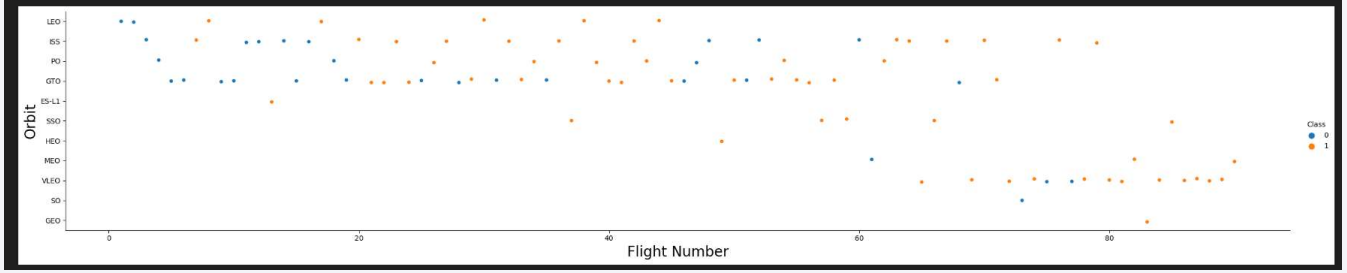
Explanations:

- The orbit types with the highest success rate (100%) are: S-l1, GEO, SSO and HEO.
- The orbit types with the lowest success rates are SO (0%) and GTO (51%).

EDA with Data Visualization - Task 4

④ Task 4: Flight Number vs. Orbit Type (Scatter Plot)

```
# 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 = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Legend:

- Class 0 (blue) represents unsuccessful launches.
- Class 1 (orange) represents successful launches.

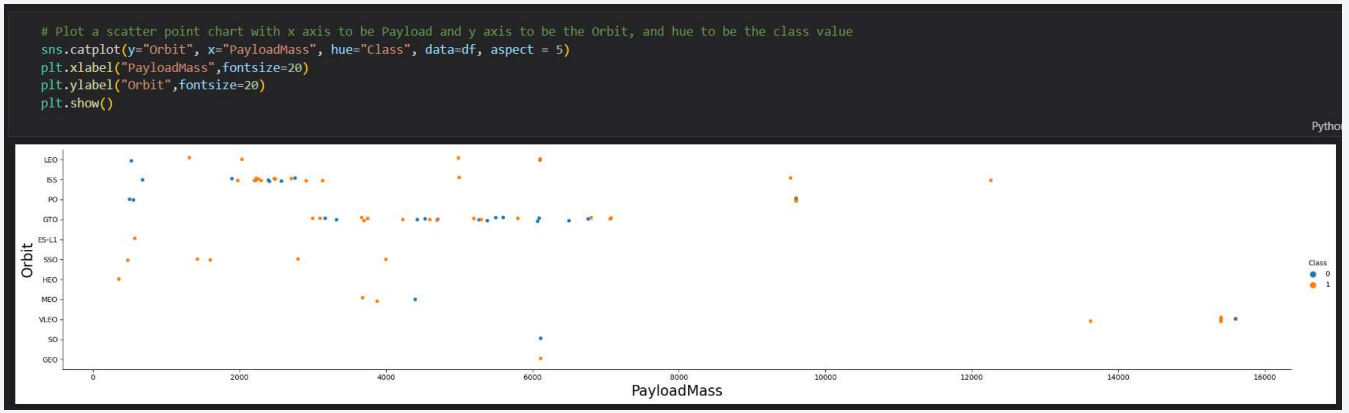
Explanations:

- As presented in the task before, we can see the most successful orbit types S-I1, GEO, SSO and HEO with their attempts and time sequence.
- The orbit type SO with the lowest success of 0% had only 1 attempt.
- We can see that orbit type VLEO launched its first rockets very late in the process but then with a relatively high success rate.

EDA with Data Visualization - Task 5

5 Task 5: Payload Mass vs. Orbit Type (Scatter Plot)

```
# 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("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Legend:

- Class 0 (blue) represents unsuccessful launches.
- Class 1 (orange) represents successful launches.

Explanations:

- For orbit type LEO we can see that the success rate increases with the payload mass.
- For orbit type GTO no pattern can be identified since the outcomes are independent from the payload mass.

EDA with Data Visualization - Task 6

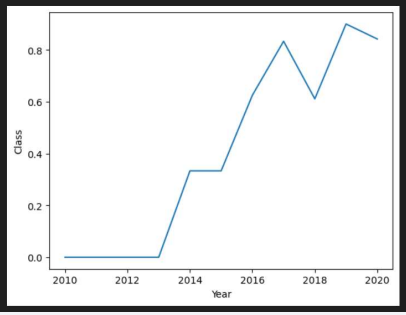
6 Task 6: Year vs. Success Rate (Line Chart)

```
# Add Success-Rate do Dataframe
data = data.groupby('Year')['Class'].mean().reset_index()
data
```

	Year	Class
0	2010	0.000000
1	2012	0.000000
2	2013	0.000000
3	2014	0.333333
4	2015	0.333333
5	2016	0.625000
6	2017	0.833333
7	2018	0.611111
8	2019	0.900000
9	2020	0.842105

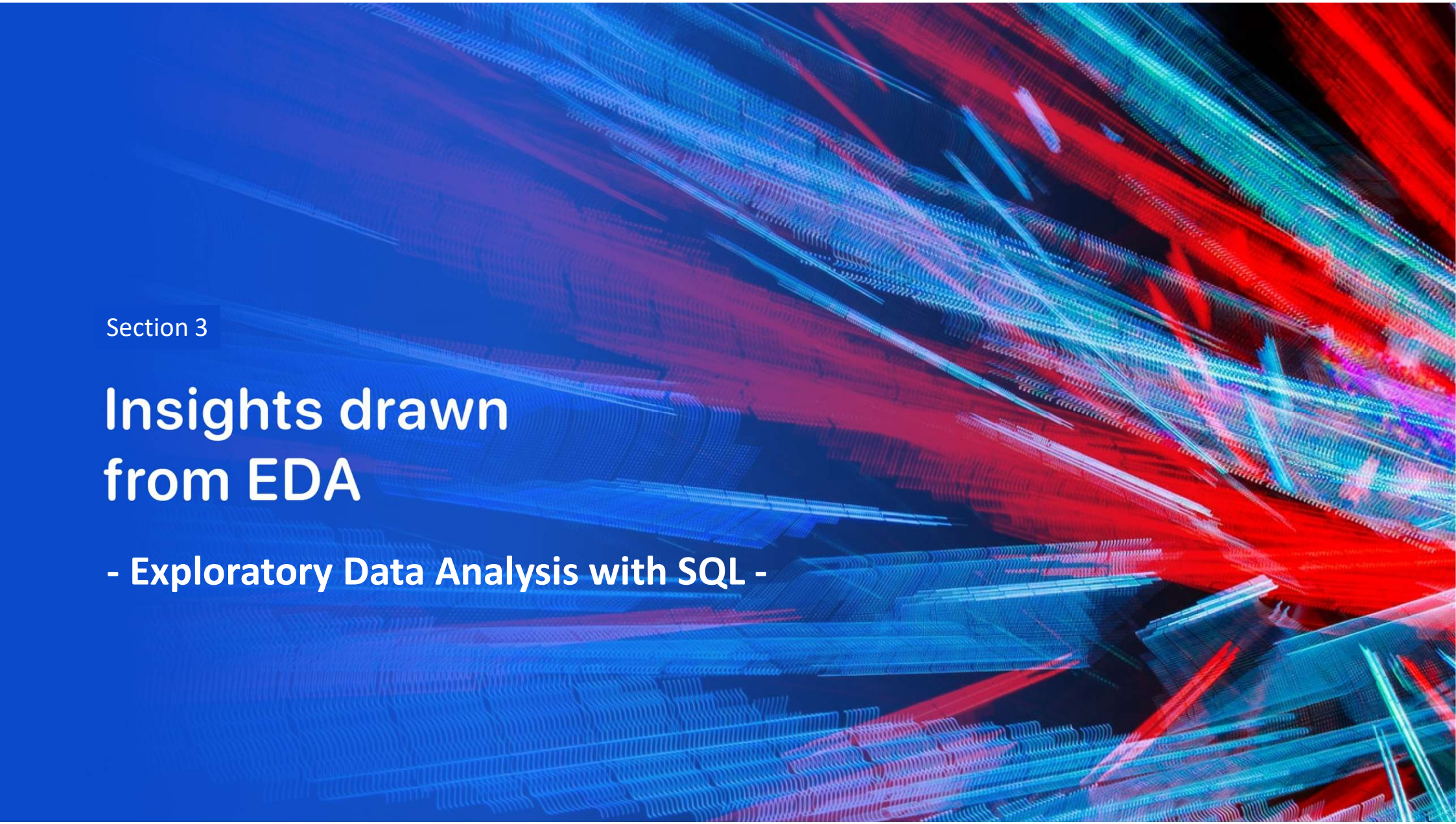
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(data=data, x="Year", y="Class")
```

```
<AxesSubplot:xlabel='Year', ylabel='Class'>
```



Explanations:

- The successrate increase from 2013 on until 2017.
- In 2018 the success rate decreased compared to the prior year and increase in 2019 again to their highest level in history of 90%.


Section 3
Insights drawn from EDA
- Exploratory Data Analysis with SQL -

EDA with SQL - Task 1

1. Task 1: Display the names of the unique launch sites in the space mission

Query:

```
%%sql
SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

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

Explanations:

- Inside a table, a column may contain duplicate values.
- The SELECT DISTINCT statement is used to return only distinct (different) values.

Result:

- There are four different launch sites in the dataset (CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E).
- With regard to the location of the launch sites, please refer to Section 3 of this presentation.

EDA with SQL - Task 2

② Task 2: Display 5 records where launch sites begin with the string 'CCA'

Query:

```
%%sql
SELECT * FROM SPACEXTBL
    ORDER BY LAUNCH_SITE
    LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

Explanations:

- Using LIMIT 5, the query gives back only the first five rows of the dataset.
- Additionally, the LIKE statement could be used in order to define the requested string properly.

Result:

- The query presents five records where the launch sites begin with the string ‘CCA’.

EDA with SQL - Task 3

3 Task 3: Display the total payload mass carried by boosters launched by NASA (CRS)

Query:

```
%%sql
SELECT "CUSTOMER", SUM(PAYLOAD_MASS__KG_) AS "TOTAL PALYOAD ASS CARRIED OUT"
    FROM SPACEXTBL
    WHERE CUSTOMER = "NASA (CRS)";
```

```
 * sqlite:///my_data1.db
Done.
```

Customer	TOTAL PALYOAD ASS CARRIED OUT
NASA (CRS)	45596

Explanations:

- By using the SELECT WHERE statement we only select the required customer (NASA (CRS)).
- We add up the Payload mass by using the SUM operator.

Result:

- The total payload mass carried by booster launched by NASA CRS amounts to 45,596 kg.

EDA with SQL - Task 4

4 Task 4: Display average payload mass carried by booster version F9 v1.1

Query:

```
%%sql
SELECT "BOOSTER_VERSION", AVG(PAYLOAD_MASS__KG_) AS "AVERAGE PALYOAD ASS CARRIED OUT"
    FROM SPACEXTBL
    WHERE BOOSTER_VERSION LIKE "F9 v1.1";
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	AVERAGE PALYOAD ASS CARRIED OUT
F9 v1.1	2928.4

Explanations:

- By using the SELECT WHERE statement we only select the required booster version (F9 v1.1).
- We calculate the average on the Payload mass by using the AVG operator.

Result:

- The average payload mass carried by booster version F) v1.1 amounts to 2,928.4 kg.

EDA with SQL - Task 5

5 Task 5: List the date when the first successful landing outcome in ground pad was achieved

Query:

```
%%sql
SELECT DATE FROM SPACEXTBL
    WHERE "LANDING _OUTCOME" = "Success (ground pad)"
    LIMIT 1;
```

```
 * sqlite:///my_data1.db
Done.
```

Date
22-12-2015

Explanations:

- Using LIMIT 1, the query gives back only the first row of the dataset.
- Additionally, the MIN(DATE) function could be used in order to select the earliest date.
- By using the SELECT WHERE statement we only select rows with where landing outcome was successful.

Result:

- The first successful landing outcome in ground pad was achieved on 22 December 2015.

EDA with SQL - Task 6

6 Task 6: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

Query:

```
%%sql
SELECT "BOOSTER_VERSION" FROM SPACEXTBL
    WHERE "LANDING _OUTCOME" = "Success (drone ship)"
    AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

```
 * sqlite:///my_data1.db
Done.
```

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

Explanations:

- By using the WHERE AND statement, we combine several conditions in our query.
- Hence, we can select the booster versions with a successful landing outcome and payload mass greater than 4000 kg but less than 6000 kg.

Result:

- 4 booster version fulfill the requirements (F9 FT B1022, F9 FT B1026, F9 FT B1021.2 and F9 FT B1031.2).

EDA with SQL - Task 7

7 Task 7: List the total number of successful and failure mission outcomes

Query:

```
%%sql
SELECT "MISSION_OUTCOME", COUNT("MISSION_OUTCOME") AS "TOTAL NUMBER OF OUTCOMES"
    FROM SPACEXTBL
    GROUP BY "MISSION_OUTCOME";
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	TOTAL NUMBER OF OUTCOMES
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Explanations:

- By using the SELECT COUNT statement, we count and group the different mission outcomes in our query.
- Hence, we can create a table grouped by the mission outcomes.

Result:

- There are 3 different mission outcomes (Failure, Success and Success (payload status unclear).

EDA with SQL - Task 8

8 Task 8: List the names of the booster versions which have carried the maximum payload mass. Use a subquery

Query:

```
%%sql
SELECT "BOOSTER_VERSION", SUM(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
    WHERE (SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
    GROUP BY (BOOSTER_VERSION)
    ORDER BY SUM(PAYLOAD_MASS__KG_) DESC
;
```

```
 * sqlite:///my_data1.db
Done.
```

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

Explanations:

- By using the SELECT statement in combination with the SUM operator, GROUP BY statement and ORDER BY statement, we can group and order the booster versions by maximum payload mass..

Result:

- As presented in the query on the left side, there are many booster versions with a maximum payload mass of 15,600 kg.

Please refer to the original notebook to see the complete list

EDA with SQL - Task 9

9 Task 9: 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

Query:

```
%%sql
SELECT "DATE", "BOOSTER_VERSION", "LANDING _OUTCOME", "LAUNCH_SITE" FROM SPACEXTBL
    WHERE "LANDING _OUTCOME" = "Failure (drone ship)"
    LIMIT 2;
```

```
 * sqlite:///my_data1.db
Done.
```

Date	Booster_Version	Landing _Outcome	Launch_Site
10-01-2015	F9 v1.1 B1012	Failure (drone ship)	CCAFS LC-40
14-04-2015	F9 v1.1 B1015	Failure (drone ship)	CCAFS LC-40

```
WHERE "LANDING _OUTCOME" = "Success" OR "LANDING _OUTCOME" = "Success (drone ship)" OR "LANDING _OUTCOME" = "Success (ground pad)"
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such function: YEAR
[SQL: SELECT * FROM SPACEXTBL  WHERE YEAR("DATE")=2015]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Explanations:

- By using the SELECT WHERE statement, we select the records with a failure landing outcome and group the different mission outcomes in our query.
- Additionally, we could use a second condition (AND YEAR(DATE) = '2015') in order to limit our query to the year 2015.

Result:

- There are 2 records with the required conditions.

EDA with SQL - Task 10

10 Task 10: Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order

Query:

```
SELECT * FROM SPACEXTBL
    WHERE (Date > '06-04-2010' AND DATE < '03-20-2017')
        AND(landing__outcome='Success'
            OR landing__outcome='Success (ground pad)'
            OR landing__outcome='Success (drone ship)'
            )
    ORDER BY Date DESC
    ;
```

SELECT * FROM SPACEXTBL WHERE (Date ... Ausführungszeit: 0.008 s

Ergebnismenge 1 Suchen

DATE	TIME__UTC_	BOOSTER_VERSION	LAUNCH_SITE	PA
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	S
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Ir
2016-08-14	05:26:00	F9 FT B1026	CCAFS LC-40	J(
2016-07-18	04:45:00	F9 FT B1025.1	CCAFS LC-40	S
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	TI

Explanations:

- By using the SELECT WHERE (DATE) statement and the ORDER BY DATE DESC, we order the records with a successful landing outcomes in a descending order.

Result:

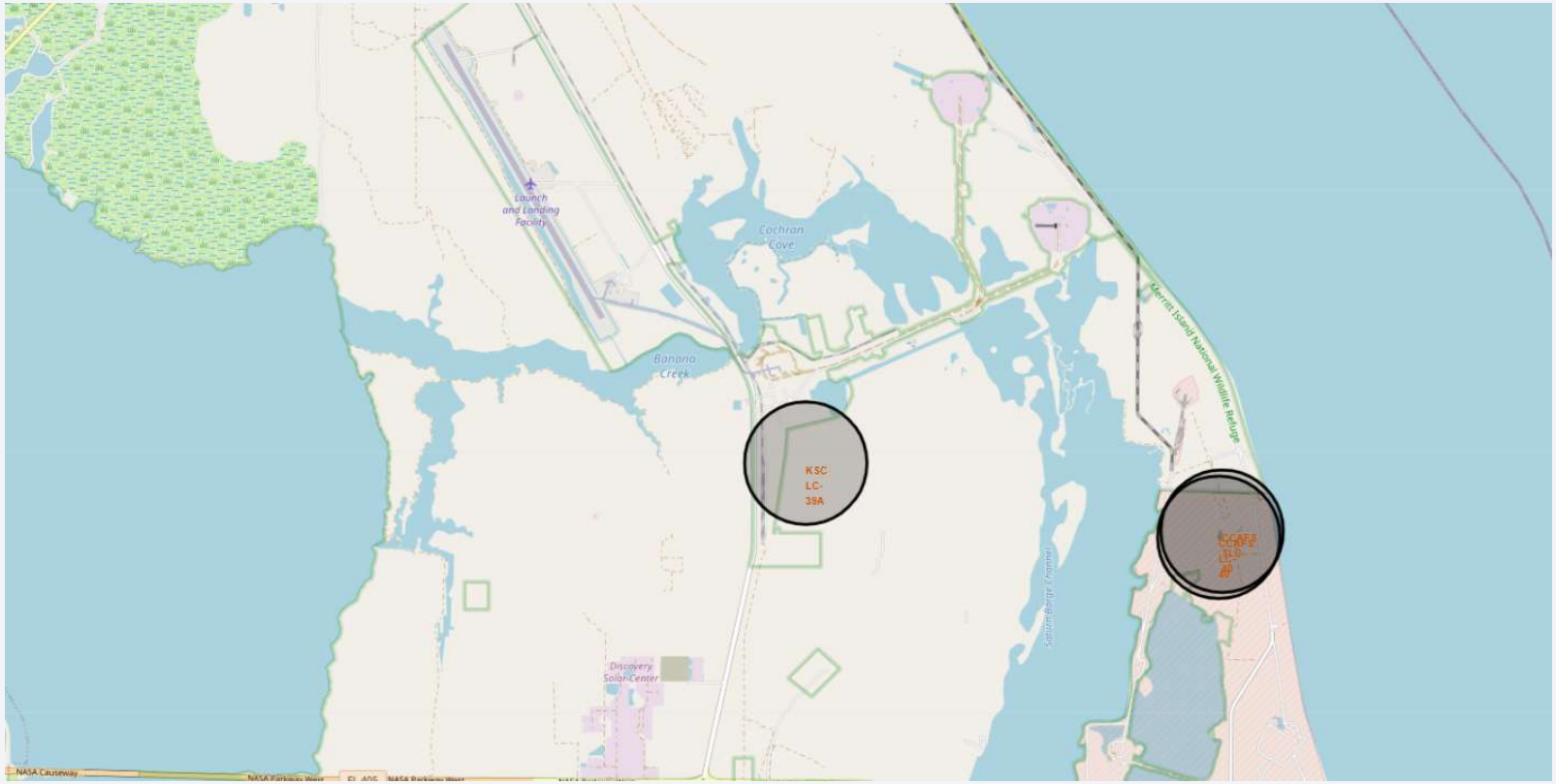
- Please refer to the results in the table on the left side.


Section 3
Launch Sites
Proximities Analysis

SpaceX launch sites - Locations (Screenshot 1)

- As presented on the folium map on the left side, there are in total four SpaceX launch sites.
- All launch sites are in very close proximity to the coast of the United States of America.
- Launch site VAFB SLC-4E is on the west cost (California).
- The Launch sites CCAFS LC-40, CCAFS SLC-40 and KSC LC-39A are very close to each other on the east coast (Florida).
- On the next page a closer screenshot presents the locations of the launch sites on the east cost.

SpaceX launch sites - Locations (Screenshot 1)

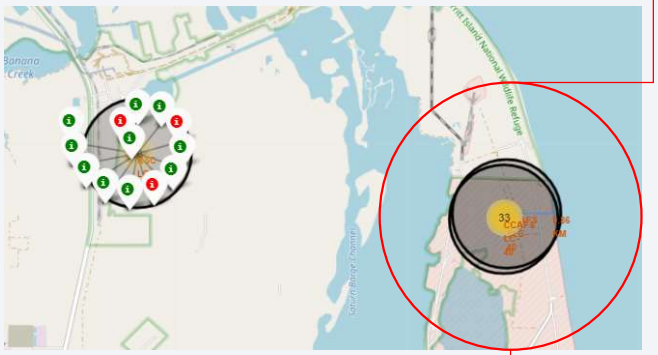


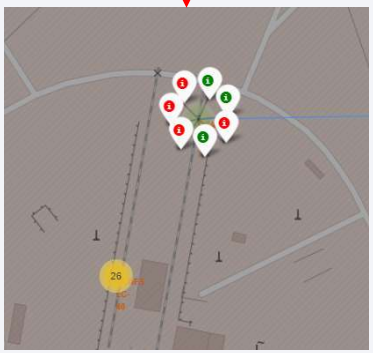
- The Launch sites CCAFS LC-40, CCAFS SLC-40 and KSC LC-39A are very close to each other on the east coast.

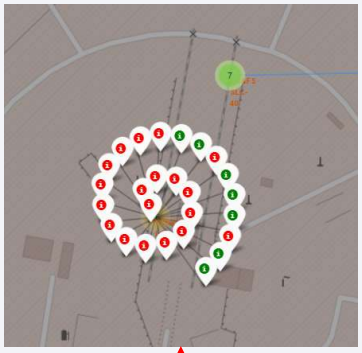
SpaceX launch sites - Success Rate (Screenshot 2)

- As we can see, there were in total 56 landings (10 landings on the launch sites located on the west coast and 46 landings on the east coasts).
- The next slide presents a more detailed view including the number of successful (green) and not successful (red) landings.

SpaceX launch sites - Success Rate (Screenshot 2)

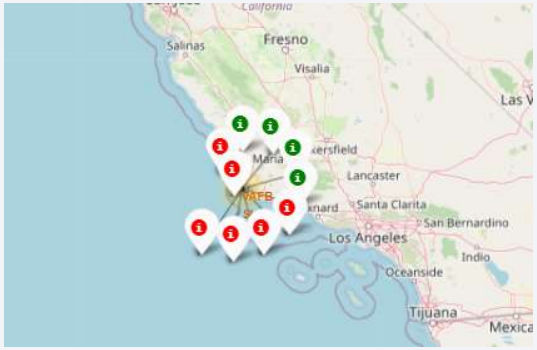






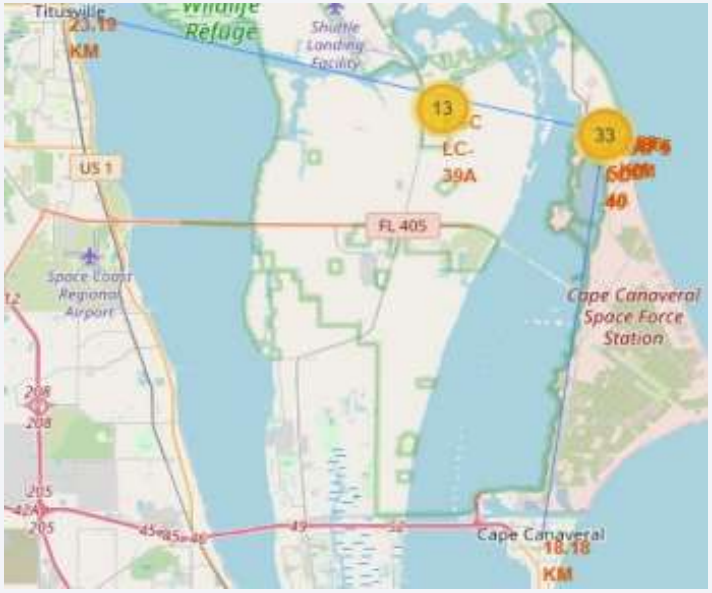
- East Coast (KSC LC-39A):
  - 10 Successful Landings
  - 3 failed landings
- East Coast (CCAFS SLC-40):
  - 3 Successful Landings
  - 4 failed landings
- East Coast (CCAFS LC-40):
  - 7 Successful Landings
  - 19 failed landings

SpaceX launch sites - Success Rate (Screenshot 2)

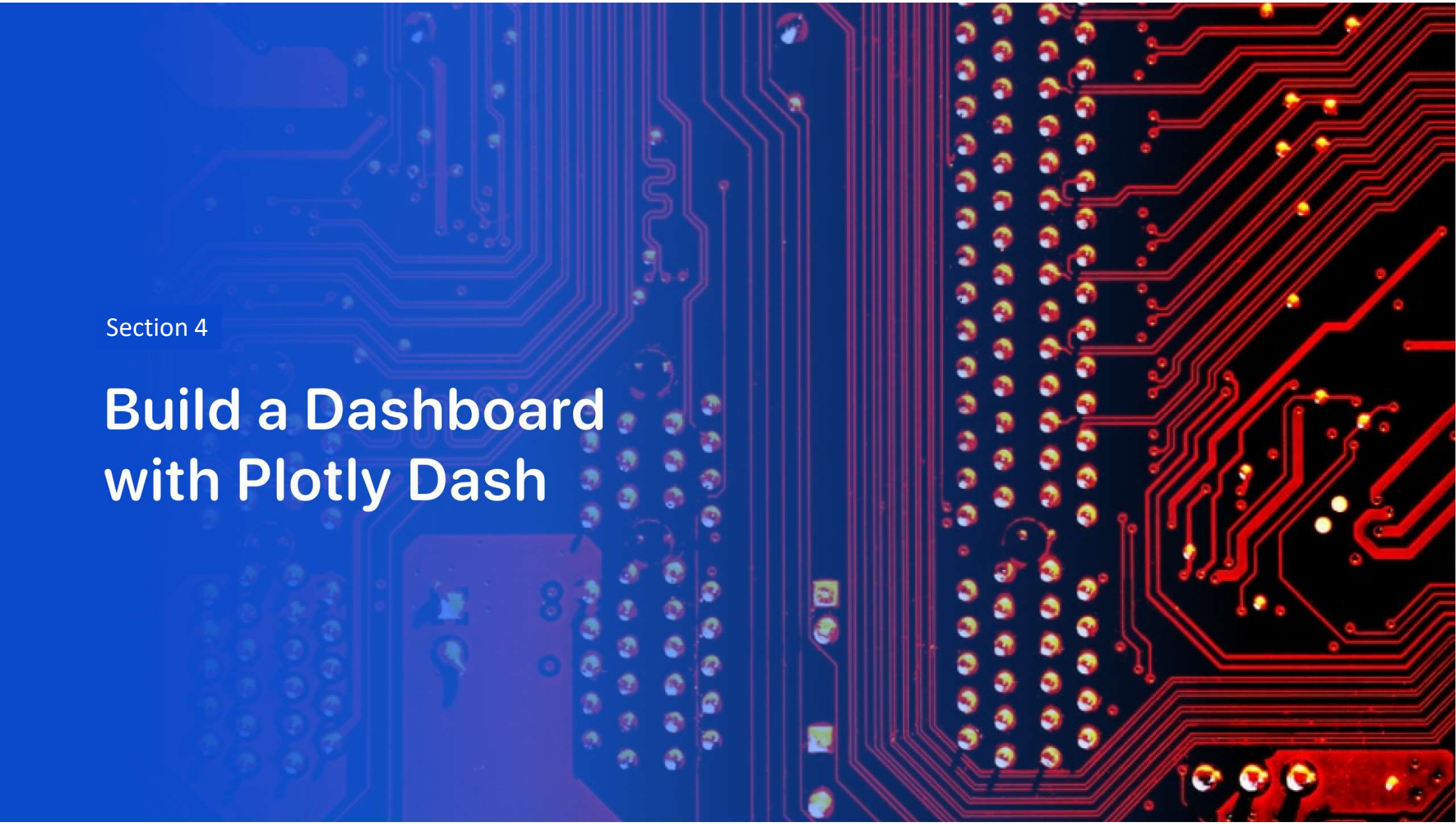


- West Coast (VAFB SLC-4E):
  - 4 Successful Landings
  - 6 failed landings

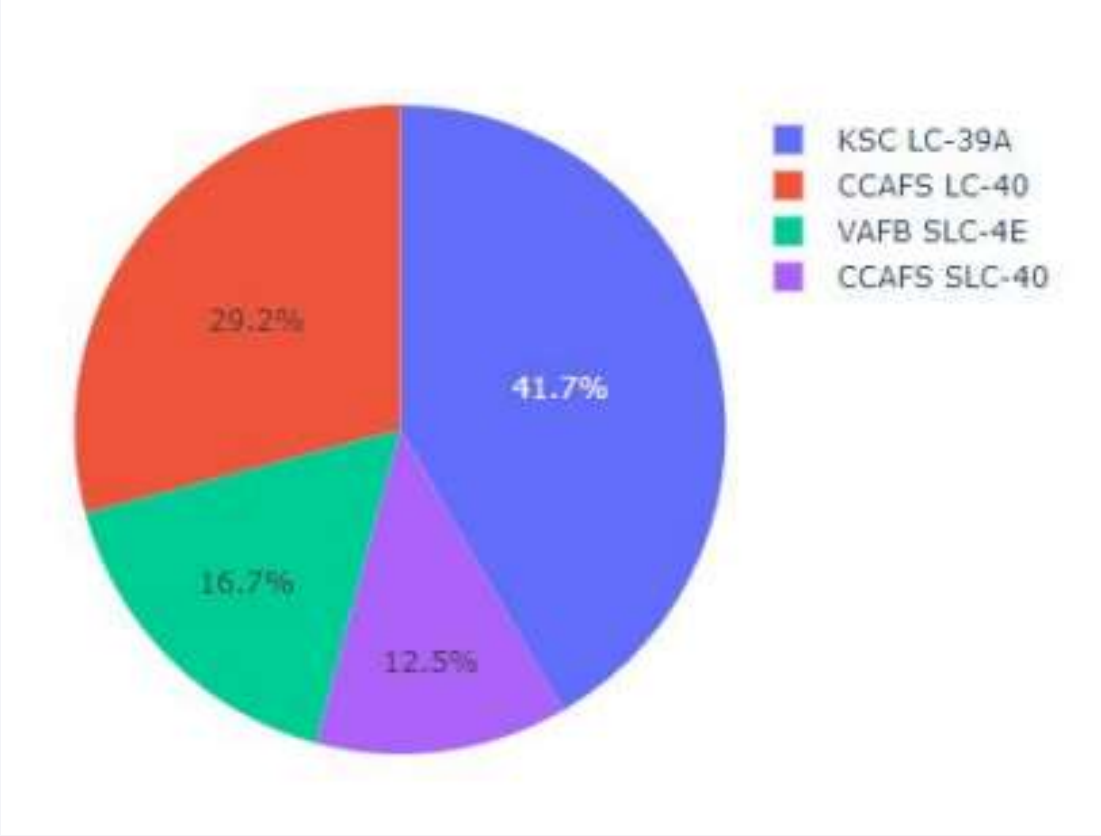
SpaceX launch sites - Distance (Screenshot 3)



- It can be found that the launch site is close to railways and highways for transportation of equipment or personnel and is also close to coastline and relatively far from the cities so that launch failure does not pose a threat.


Section 4
Build a Dashboard with Plotly Dash

Launch success count for all sites (Task 1)

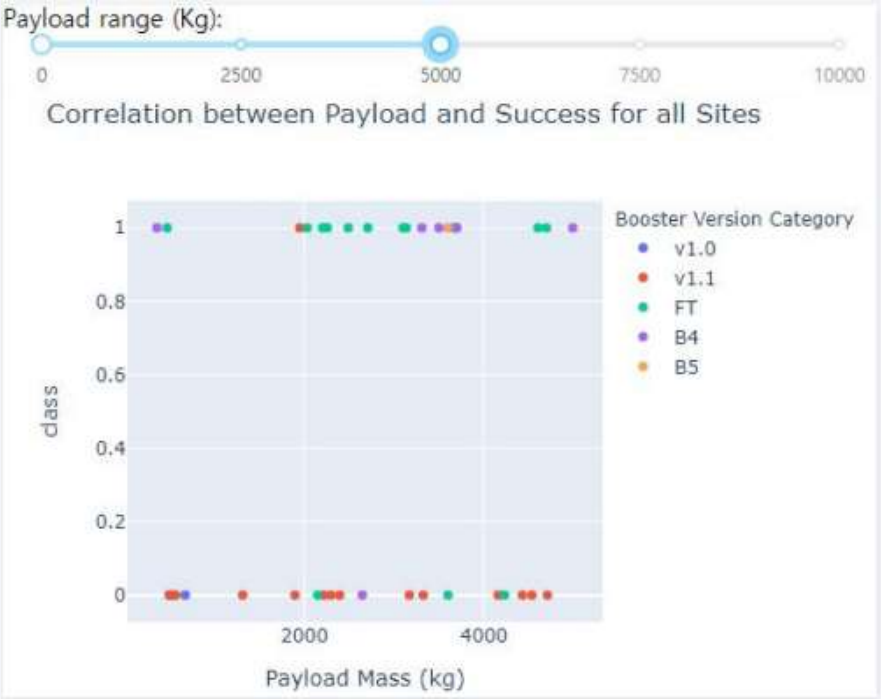


- The pie chart on the left side presents the most launch success among all sites.
- KSC LC-39A has the most launch success with 41.7%
- The launch site with the lowest success is CCAFS SLC-40 with 12.5%

Launch Site with Highest Launch Success Ratio (Task 2)

- The pie chart on the left side presents the site with the highest success rate.
- KSLC-39A has the highest success rate with 10 landing successes (76.9%) and 3 landing failures (23.1%).

Payload vs. Launch Outcome Scatter Plot for all sites (Task 3)

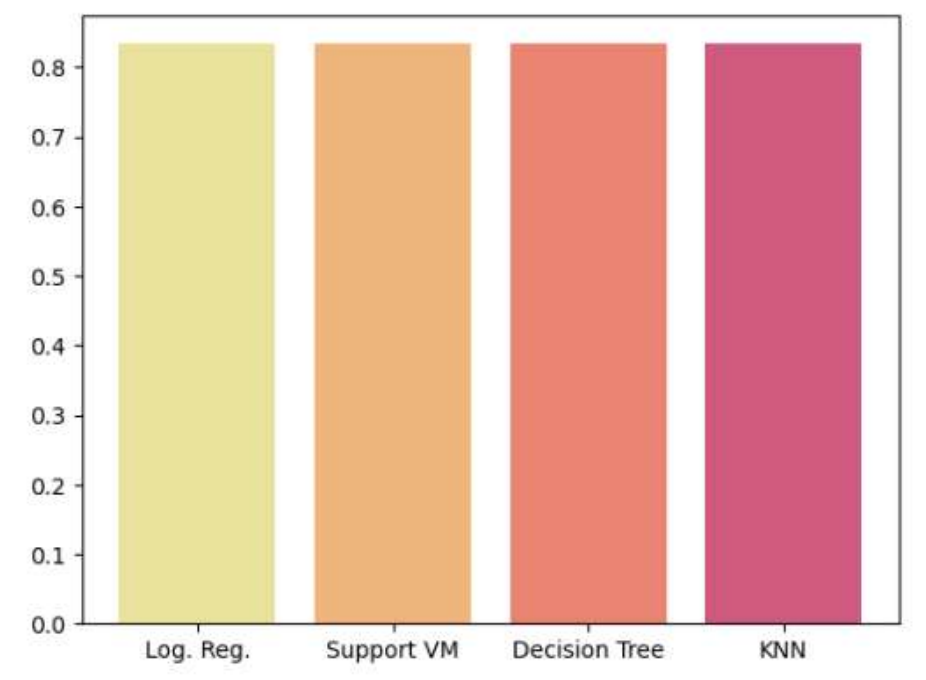


- These figures show that the launch success rate (class 1) for low weighted payloads (0-5000 kg) is higher than that of heavy weighted payloads (5000-10000 kg).

Section 5

Predictive Analysis (Classification)

Classification Accuracy

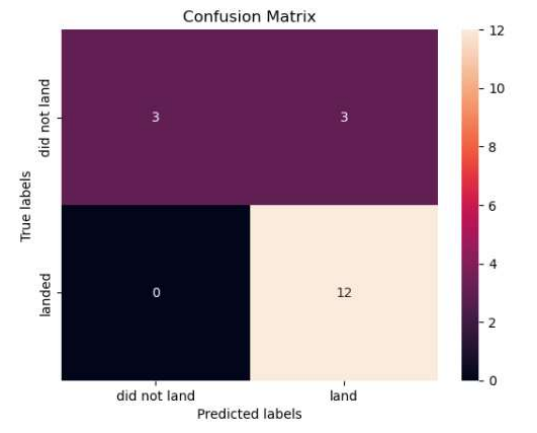


	Log. Reg.	Support VM	Decision Tree	KNN
Accuracy Score	0.833333	0.833333	0.833333	0.833333

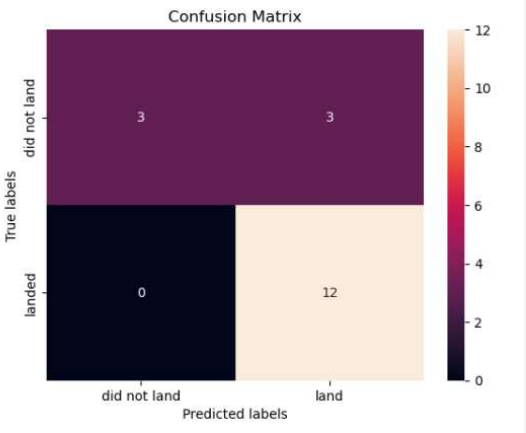
- We applied four methods to calculate an accuracy score
  1) Logistic Regression
  2) Support Vector Machine
  3) Decision Tree Classifier
  4) K nearest neighbors
- Applying these methods on the test set, the accuracy of all models is the same at 83.33%.
- Please refer to the table for the accuracy scores and the bar chart on the left side for visualization purposes.

Confusion Matrix

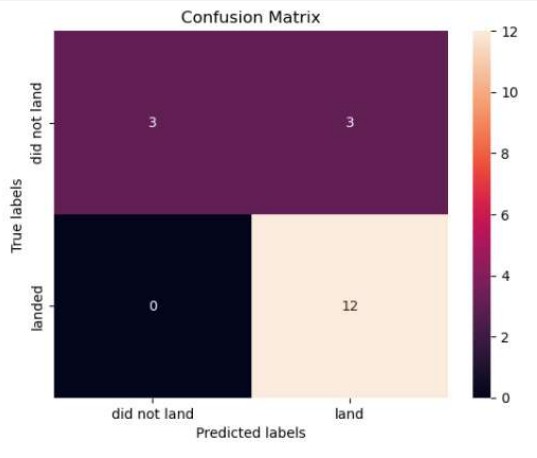
Logistic Regression



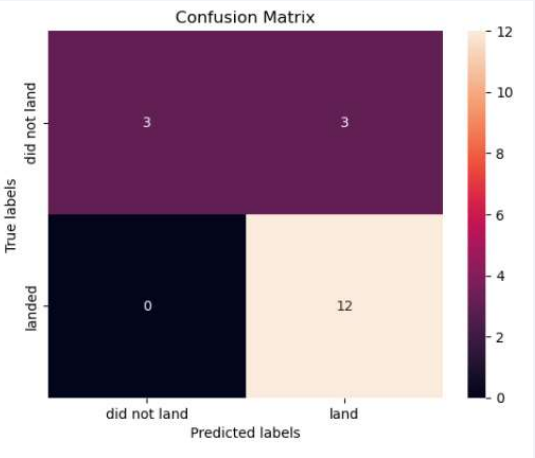
Decision Tree Classifier



Support Vector Machine



K neirest neighbors



- All confusion matrix come to the same result, since all applied methods performed the same for the test set.
- The models predicted 12 successful landings when the true label was successful and 3 failed landings when the true label was failure. But there were also 3 predictions that said successful landings when the true label was failure (false positive).
- Overall, these models predict successful landings.

Results

- Orbit types SSO, HEO, GEO and ES-L1 have the highest success rate (100%).
- KSLC-39A has the highest number of launch successes and the highest success rate among all sites.
- As the number of flights increased, the success rate increased, and recently it has exceeded 80%.
- The launch site is close to railways, highways, and coastline, but far from cities.
- The launch success rate of low weighted payloads is higher than that of heavy weighted payloads.
- In this dataset, all models have the same accuracy (83.33%).

Appendix

Github URLs:

- Final Data Science Capstone Project - Week 1 - Lab 1 - SpaceX Data Collection API.ipynb
- Final Data Science Capstone Project - Week 1 - Lab 2 - SpaceX Webscraping.ipynb
- Final Data Science Capstone Project - Week 1 - Lab 3 - SpaceX Data Wrangling.ipynb
- Final Data Science Capstone Project - Week 2 - Lab 2 - SpaceX EDA with Data Visualization.ipynb
- Final Data Science Capstone Project - Week 2 - Lab 1 - SpaceX EDA with SQL.ipynb
- Final Data Science Capstone Project - Week 3 - Lab 1 - Interactive Visual Analytics.ipynb
- Final Data Science Capstone Project - Week 3 - Lab 2 - Ploty Dash-App.py
- Final Data Science Capstone Project - Week 4 - Lab 1 - Machine Learning Prediction.ipynb

Coursera Applied Data Science Capstone Course URL:

- https://www.coursera.org/learn/ibm-data-analyst-capstone-project

Thank you!