Winning Space Race with Data Science

Leonardo Assi
October 2nd, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Building an interactive map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analysis

Introduction

Project background and context:

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars, while other providers cost upward of 165 million dollars each. Much of the savings is because SpaceX can reuse the first stage. It means that being able to predict the success of the first stage launch represents an important capacity to reduce the total cost of rocket launches.

Problems we want to find answers:

The project aims to determine the factors that drive a successful rocket landing. Our purpose is to predict if the first stage of the SpaceX Falcon 9 rocket will land successfully and determine in what conditions the success rates are higher.

Section 1

Methodology

Methodology

- Data collection
  - Collect data though SpaceX Rest API and web scraping from Wikipedia
- Perform data wrangling
  - Process data to create success/fail outcome variable
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build and evaluate models to predict landing outcomes using ML techniques like logistic regression, support vector machine (SVM), decision tree and K-nearest neighbor (KNN)

Data Collection

- Data collection is the process of collecting information on relevant variables in a predetermined, methodical way.
- Data was collected though SpaceX Rest API and web scraping from Wikipedia
  - The objective of the API data collection was to request to the SpaceX API and clean the requested data
  - The objective of the Web Scrapping data collection was to extract a Falcon 9 launch records HTML table from Wikipedia and then parse the table and convert it into a Pandas data frame

DATA COLLECTION VIA SPACEX REST API

1. Requesting rocket launch data from SpaceX API
2. Decoding the response content as a JSON file
3. Converting the JSON result into a dataframe
4. Getting information about the launches using the IDs given for each launch
5. Cleaning data using custom functions
6. Combining columns into a dictionary & constructing dataset
7. Filtering dataframe & printing results

DATA COLLECTION VIA WEB SCRAPPING

1. Getting response from HTML
2. Creating BeautifulSoup Object from response
3. Extracting all column/variable names from the HTML table header
4. Creating dictionay and appending data to keys
5. Converting dictionary to dataframe
6. Exporting dataframe to .csv

GitHub URL

Data Collection – SpaceX API

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

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

Requesting rocket launch data from SpaceX API

Decoding the response content as a JSON file

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

Converting the JSON result into a dataframe

```
data = pd.json_normalize(response.json())
```

Getting information about the launches using the IDs given for each launch

```
# Lets take a subset of our dataframe keeping only the features we want and the flight numb
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra roc
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Cleaning data using custom functions

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

Combining columns into a dictionary & constructing dataset

```
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)
```

Filtering dataframe & printing results

```
print(data_falcon9['BoosterVersion'].value_counts())
data_falcon9 = data_falcon9[data_falcon9['BoosterVersion']=='Falcon 9']
print(data_falcon9['BoosterVersion'].value_counts())
```

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
4	1	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
5	2	2012-05-22	Falcon 9	525.0	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	677.0	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	500.0	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	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

[GitHub URL](#)

Data Collection – Web Scraping

1. Getting response from HTML

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
r = requests.get(static_url)
data = r.text
```

2. Creating BeautifulSoup Object from response

```
soup = BeautifulSoup(data,"html.parser")
```

3. Extracting all column/variable names from the HTML table header

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
print(first_launch_table)
table_headers = first_launch_table.find_all('th')
print(table_headers)
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4. Creating dictionay and appending data to keys

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each val
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']=[]
```

5. Converting dictionary to dataframe

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

6. Exporting dataframe to .csv

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

	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
...	...	...	...	...	...	...	...	...	...	...	...
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	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	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

[GitHub URL](#)

Data Wrangling

- Data wrangling is the process of transforming and structuring data from one raw form into a desired format with the intent of improving data quality and making it more consumable and useful for analysis.

Loading data → Creating dataframe → Cleaning data → Exporting data

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

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

```
# Calculate the mean value of PayloadMass column
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean())
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

EDA with Data Visualization

- A scatter plot was used to visualize the relationship between Flight Number and Launch Site
- A scatter plot was plotted to visualize the relationship between Payload and Launch Site
- A bar chart was used to visualize the relationship between success rate of each orbit type
- A scatter plot was used to visualize the relationship between Flight Number and Orbit type
- A scatter plot was used to visualize the relationship between Payload and Orbit type
- A line chart was used to visualize the launch success yearly trend

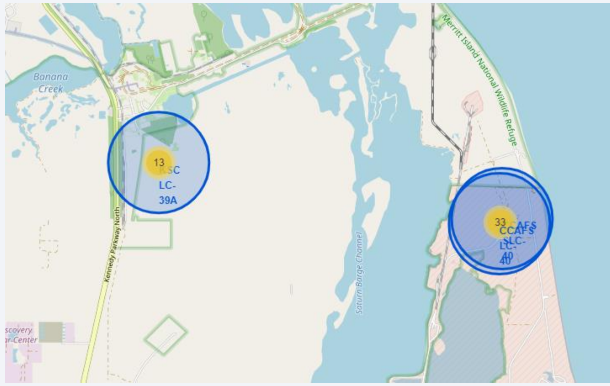
EDA with SQL

- A series of SQL queries were performed to obtain information from the Spacex DataSet:
    - Find the names of the unique launch sites in the space mission
    - Find 5 records where launch sites begin with the string 'KSC'
    - Find the total payload mass carried by boosters launched by NASA (CRS)
    - Find average payload mass carried by booster version F9 v1.1
    - List the date where the succesful landing outcome in drone ship was achieved
    - List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
    - List the total number of successful and failure mission outcomes
    - List the names of the booster_versions which have carried the maximum payload mass.
    - List the records which will display the month names, succesful landing_outcomes in ground pad,`booster versions, launch_site for the months in year 2017
    - 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

- Tasks performed using Folium:
  - Mark all launch sites on a map
  - Mark the success/failed launches for each site on the map
  - Calculate the distances between a launch site to its proximities
- The map objects created and added to the maps were:
  - **Map Marker:** used to create a mark on the map
  - **Icon Marker:** used to create an icon on the map
  - **Circle Marker:** used to create a circle on the map where marker is placed
  - **PolyLine:** used to create a line between points
  - **Marker Cluster Object:** used to simplify a map that contains many markers
  - **AntPath:** used to create an animated line between points

Examples to illustrate:



Build a Dashboard with Plotly Dash

- Plots/graphs and interactions added to the dashboard:

Type	Reason
Dropdown	used for Launch Site selection
Rangeslider	used for Payload Mass range selection
Scatter Chart	used for correlation display
Pie Chart	used for Success percentage display

Predictive Analysis (Classification)

Steps taken to build, evaluate, improve, and find the best performing classification model:

- Create a NumPy array from the column Class in data
- Standardize the data
- Split the data into training and test data
- Create a Logistic Regression object
- Calculate the accuracy of the Logistic Regression model on the test data
- Create a Support Vector Machine (SVM) object
- Calculate the accuracy of the SVM modelo on the test data
- Create a decision tree classifier object
- Calculate the accuracy of the Decision Tree model on the test data
- Create a K-Nearest-Neighbors object
- Calculate the accuracy of the KNN model on the test data
- Find the method that performs best

Results

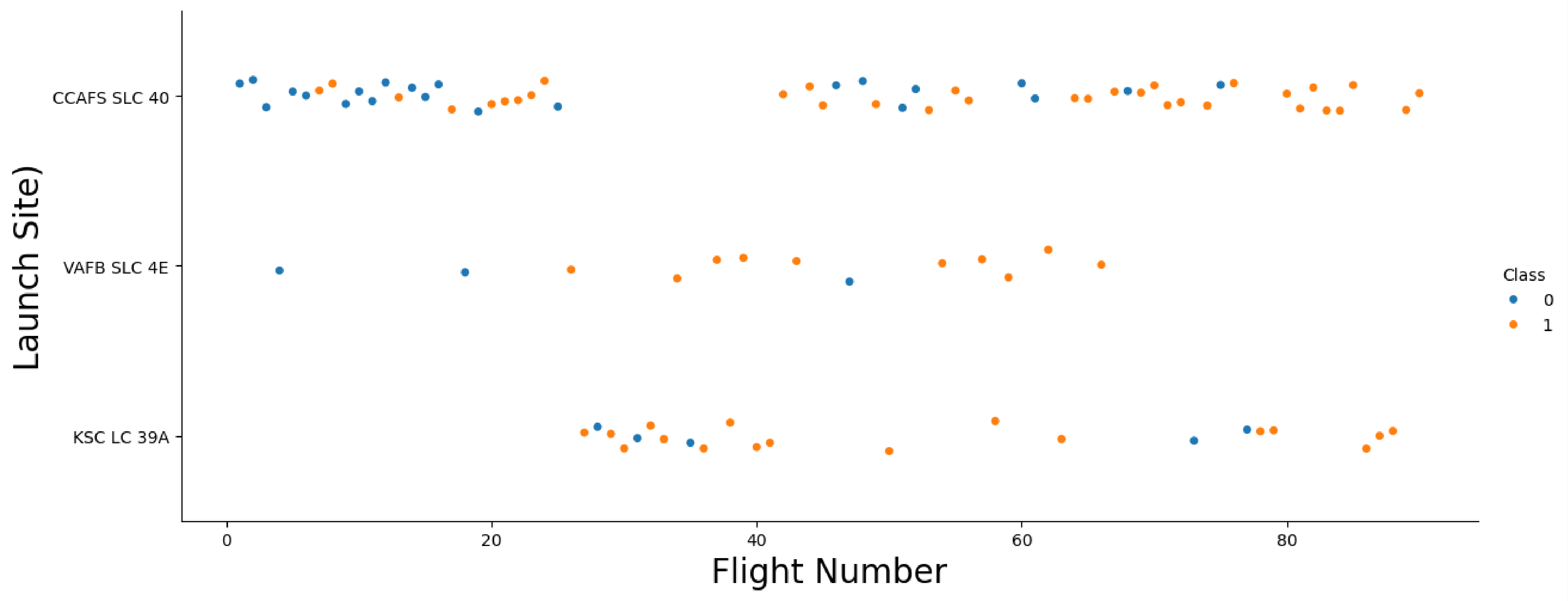
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

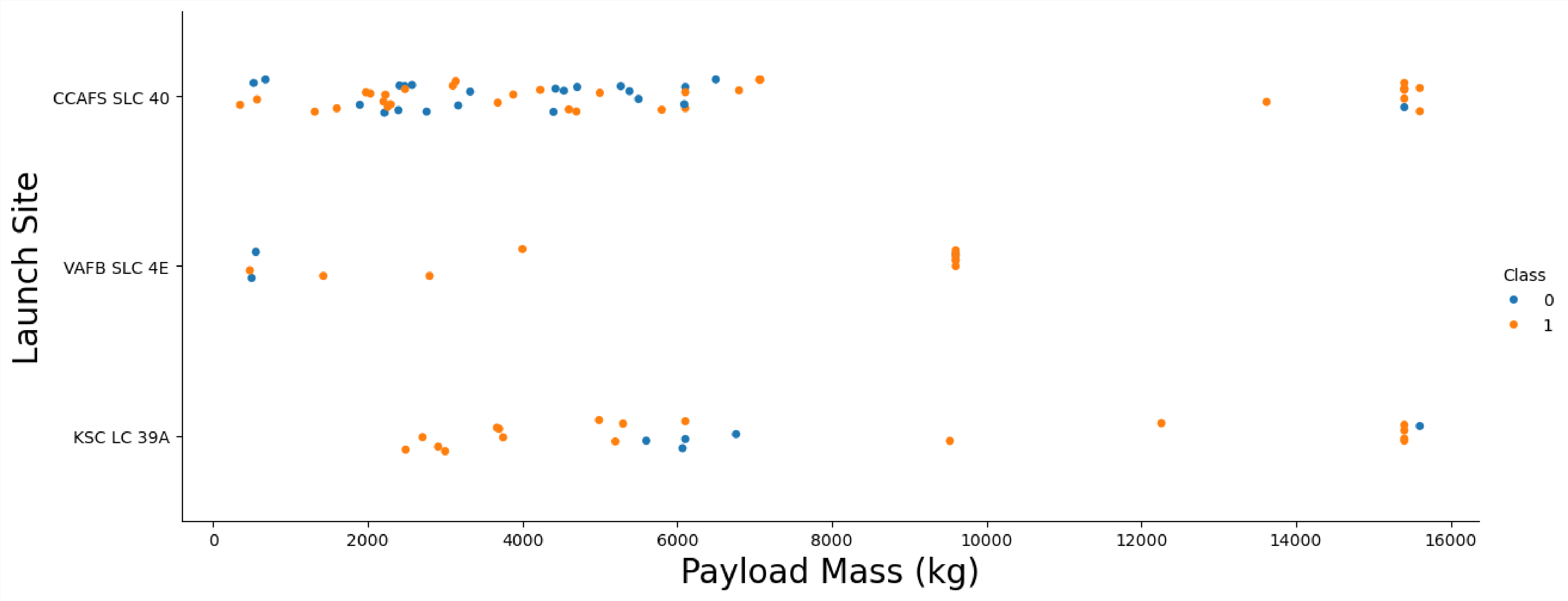
Flight Number vs. Launch Site

- The success rate increases for higher flight numbers.



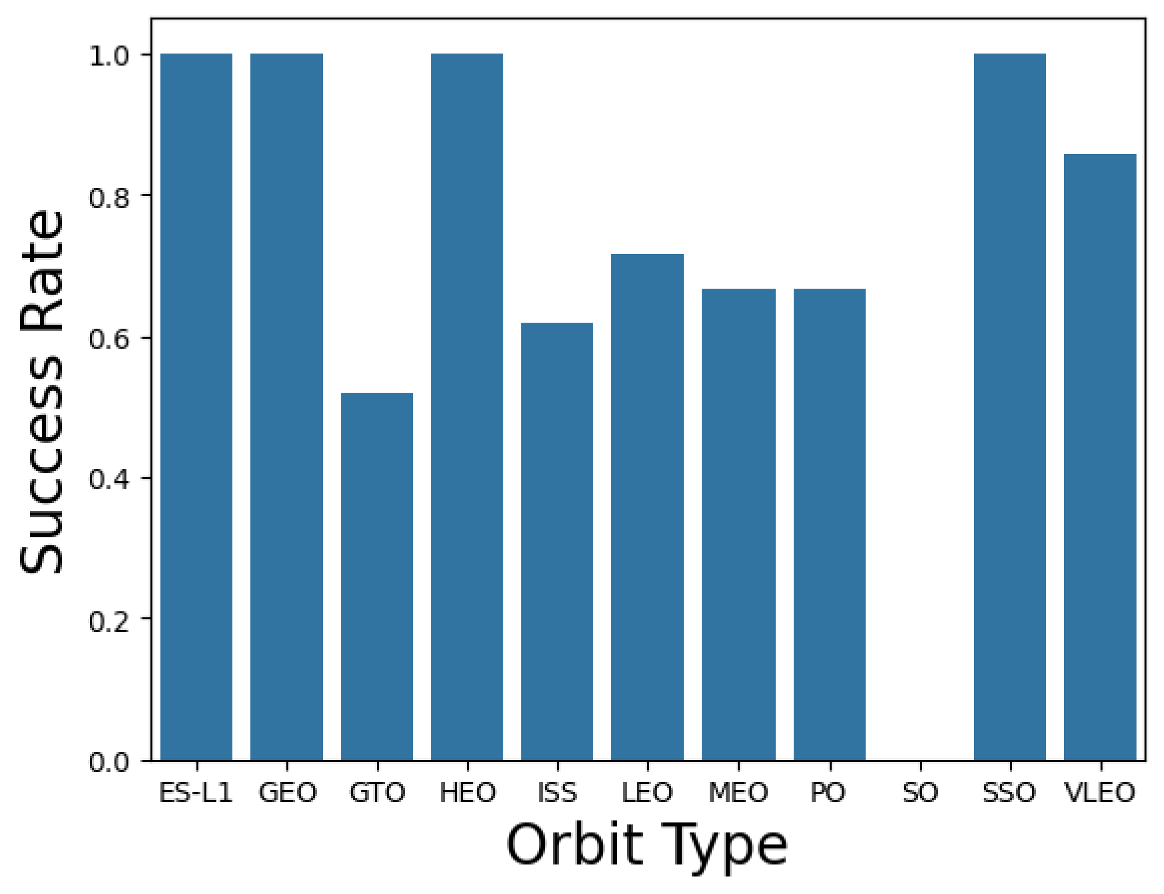
Payload vs. Launch Site

- The success rate increases for greater payload mass.

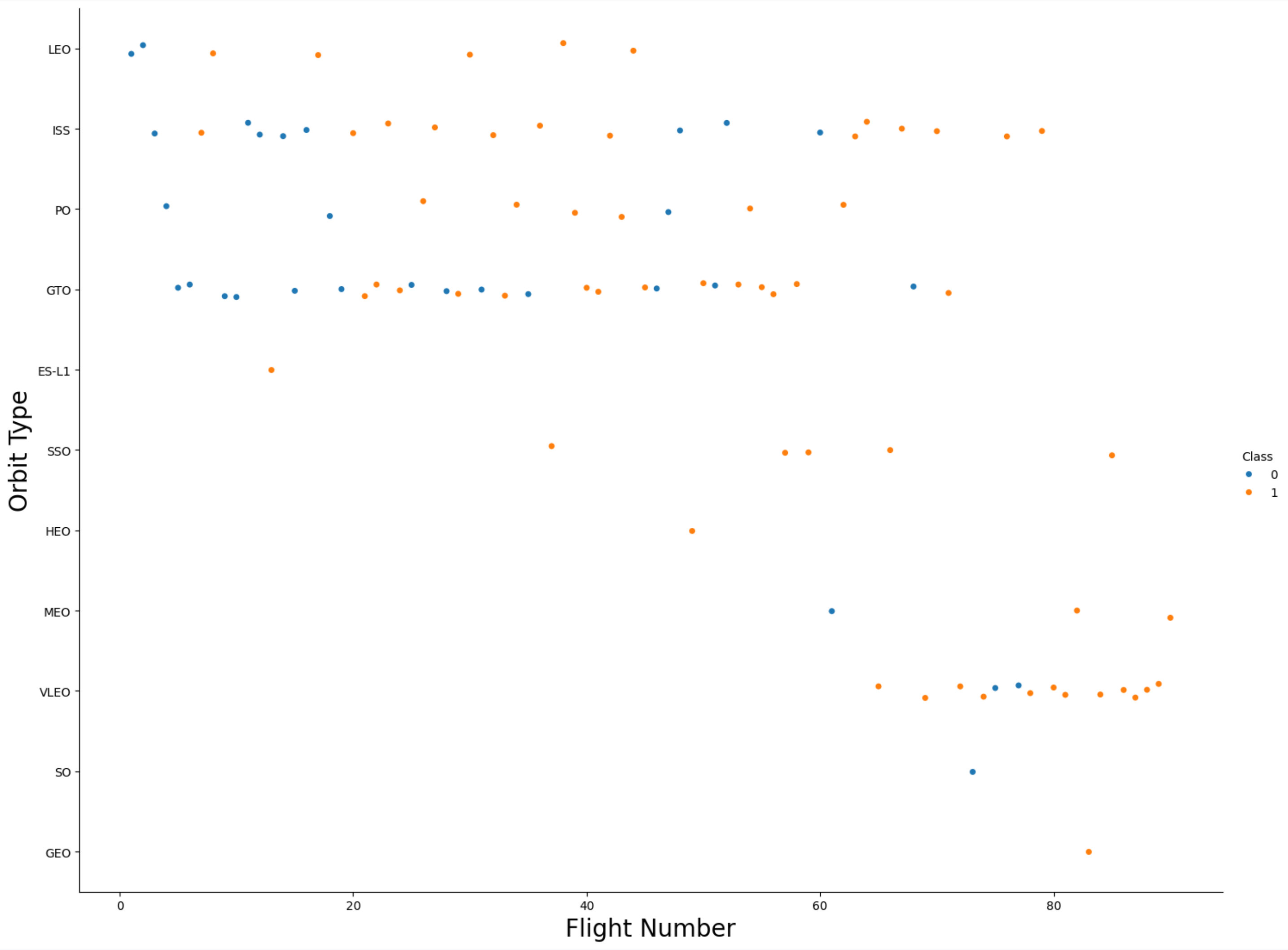


Success Rate vs. Orbit Type

- The highest success rates are related to ES-L1, GEO, HEO and SSO.



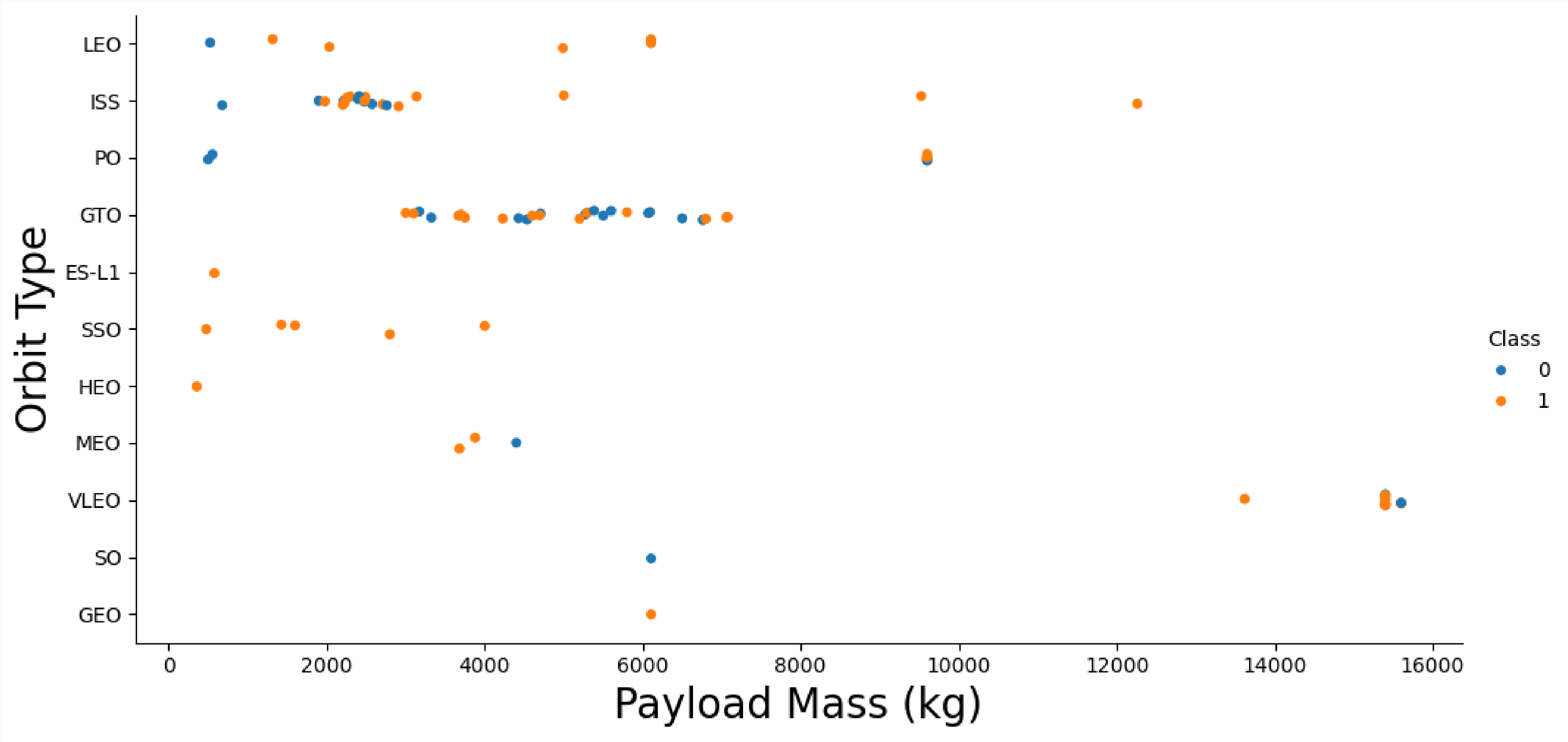
Flight Number vs. Orbit Type



- It seems that for orbit type LEO the success rate increases with flight number, which is not true for other orbite types.

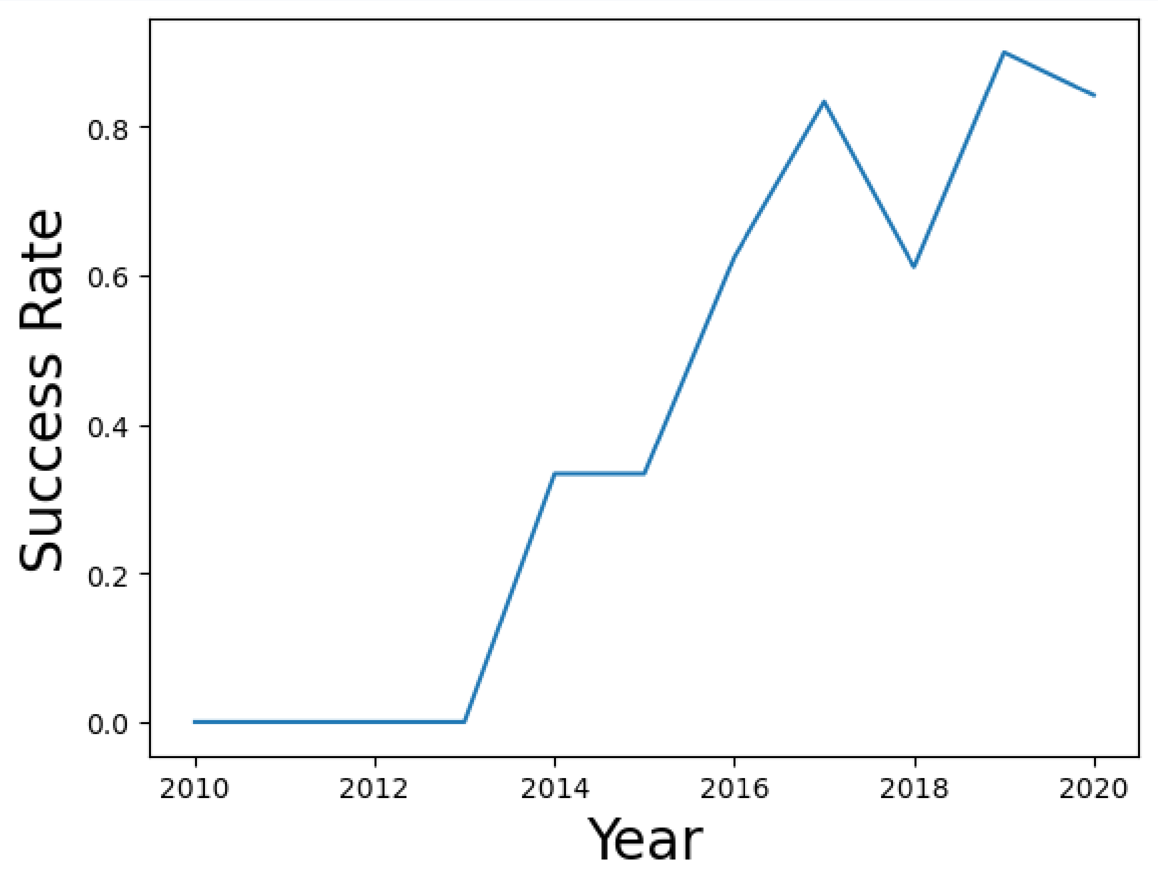
Payload vs. Orbit Type

- For orbit type LEO, greater payload mass is related to highest success rates, while for MEO it is the opposite.



Launch Success Yearly Trend

- Since 2013, success rates increase thrgouh the years, excepts for a drop in 2018.



All Launch Site Names

- The query resulted in the distinct Launch Site names

```
q = pd.read_sql('select distinct Launch_Site from spacexdata', conn)
q
```

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

Launch Site Names Begin with 'KSC'

- The query resulted in the 5 records where Launch Sites’ names start with “KSC”:

```
q = pd.read_sql("select * from spacexdata where Launch_Site like 'CCA%' limit 5", conn)
```

index	Date	Time_(UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcom
0	2010-06-04 00:00:00	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success
1	2010-12-08 00:00:00	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success
2	2012-05-22 00:00:00	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success
3	2012-10-08 00:00:00	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success
4	2013-03-01 00:00:00	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success

Total Payload Mass

- The query resulted in the total payload carried by boosters from NASA:

```
q = pd.read_sql("select sum(PAYLOAD_MASS__KG_) from spacexdata where Customer='NASA (CRS)'", conn)
```

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- The query resulted in the average payload mass carried by booster version F9 v1.1:

```
q = pd.read_sql("select avg(PAYLOAD_MASS__KG_) from spacexdata where Booster_Version='F9 v1.1'", con
n)
q
```

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- The query resulted in the date of the first successful landing outcome on drone ship:

```
q = pd.read_sql("select min(Date) from spacexdata where Landing__Outcome='Success (ground pad)'", co
nn)
q
```

min(Date)
2015-12-22 00:00:00

Successful Drone Ship Landing with Payload between 4000 and 6000

- The query resulted in the list of the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where Landing__Outcome='Success (dr
one ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000", conn)
q
```

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

Total Number of Successful and Failure Mission Outcomes

- The query resulted in the total number of successful and failure mission outcomes

```
q = pd.read_sql("select substr(Mission_Outcome,1,7) as Mission_Outcome, count(*) from spacexdata  gr
oup by 1", conn)
q
```

Mission_Outcome	count(*)
Failure	1
Success	100

Boosters Carried Maximum Payload

- The query resulted in the names of the booster which have carried the maximum Payload Mass:

```
q = pd.read_sql("select distinct Booster_Version from spacexdata where PAYLOAD_MASS__KG_ = (select m
ax(PAYLOAD_MASS__KG_) from spacexdata)", conn)
q
```

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

2015 Launch Records

- The query resulted in the list of failed landing outcomes in drone ship, their booster versions and launch site names for in year 2015:

```
q = pd.read_sql("select distinct Landing__Outcome, Booster_Version, Launch_Site from spacexdata wher
e Landing__Outcome='Failure (drone ship)'", conn)
q
```

Landing__Outcome	Booster_Version	Launch_Site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1017	VAFB SLC-4E
Failure (drone ship)	F9 FT B1020	CCAFS LC-40
Failure (drone ship)	F9 FT B1024	CCAFS LC-40

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

- The query resulted in the rank of 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:

```
q = pd.read_sql("select Landing__Outcome, count(*) from spacexdata where Date between '2011-06-04' a
nd '2017-03-20' group by Landing__Outcome order by 2 desc", conn)
q
```

Landing__Outcome	count(*)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

Launch Sites on Folium Map

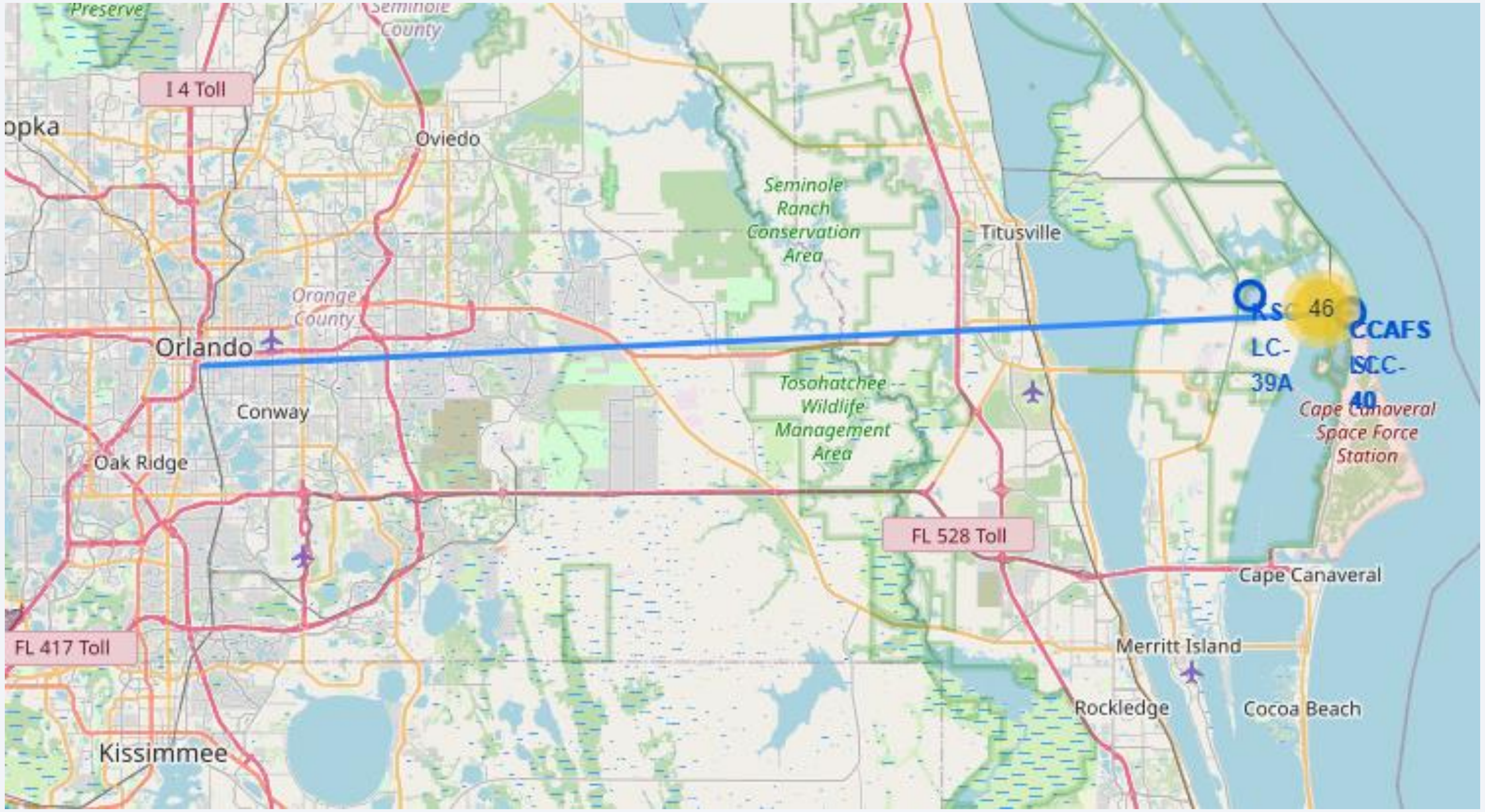
- Launch site locations on the map:

Launch Records on Folium Map

- Launch outcomes on the map:

Launch Site Distances on Folium Map

- Distance line on the map:

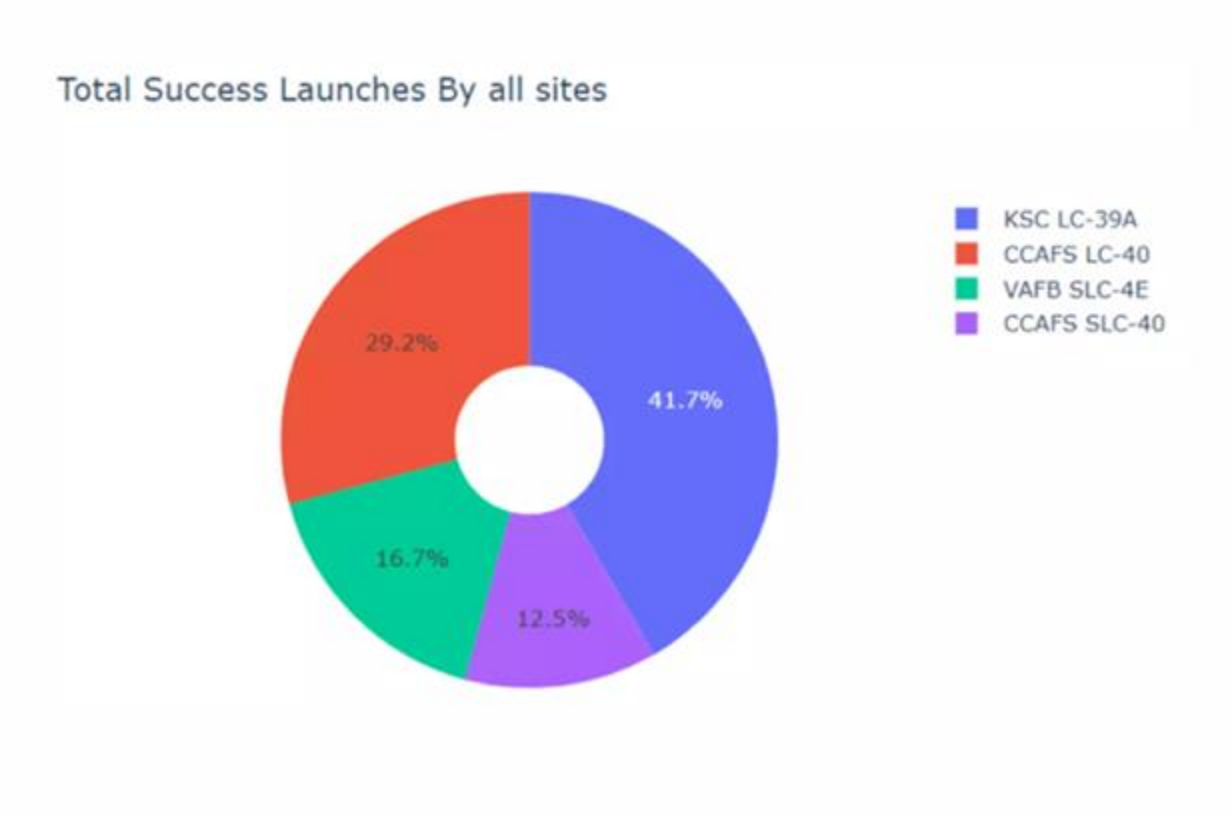


Section 4

Build a Dashboard with Plotly Dash

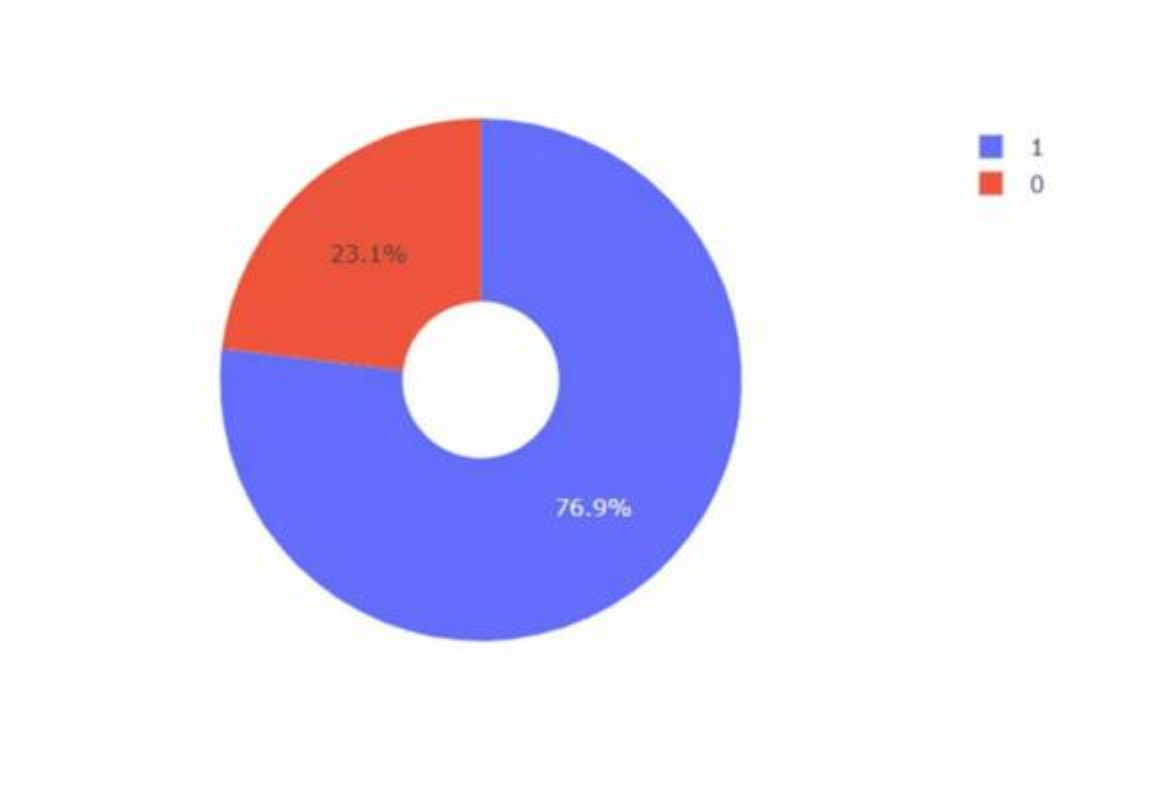
Success launches by site

- It is possible to observe by the pie chart that **KSC LC-39A** represents the most successful launches among all launch sites.



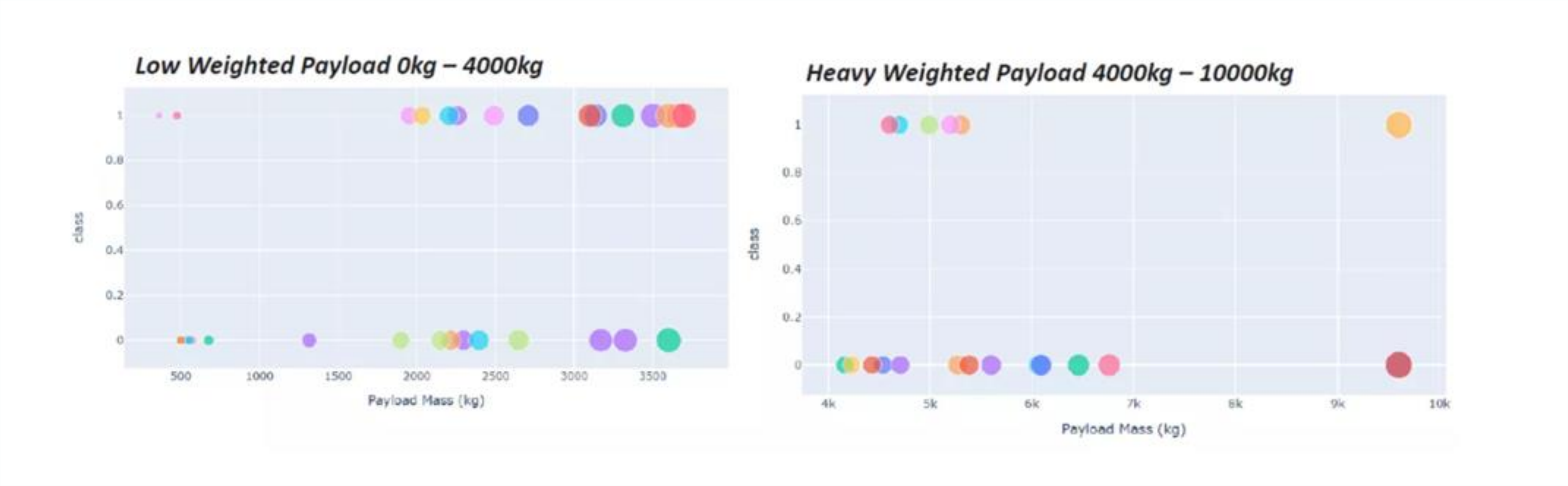
Launch site with highest success ratio

- Launch site **KSC LC-39A** achieved the highest success ratio, by reaching a 75.9% success rate and a 23.1% failure rate



Payload versus Launch Outcome for all the sites

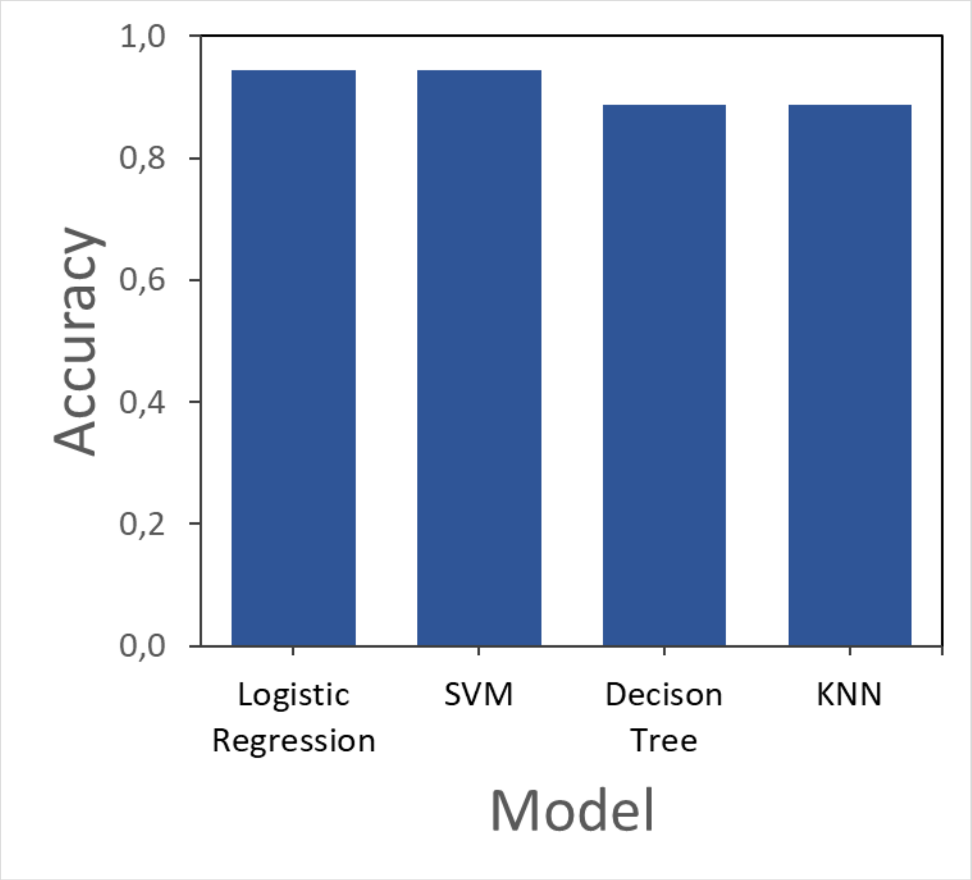
- The scatter plots show that the success rates for low weighted payloads are greater than those for have weighted payloads.



Section 5

Predictive Analysis (Classification)

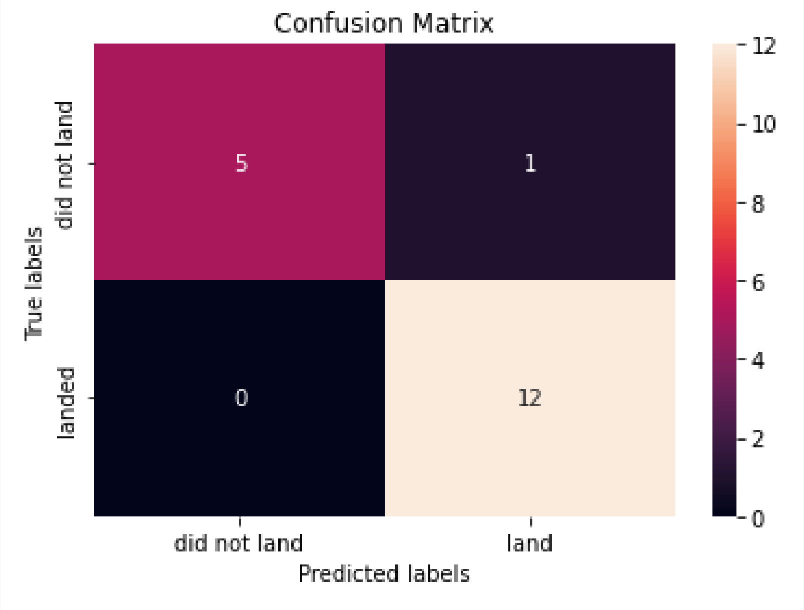
Classification Accuracy



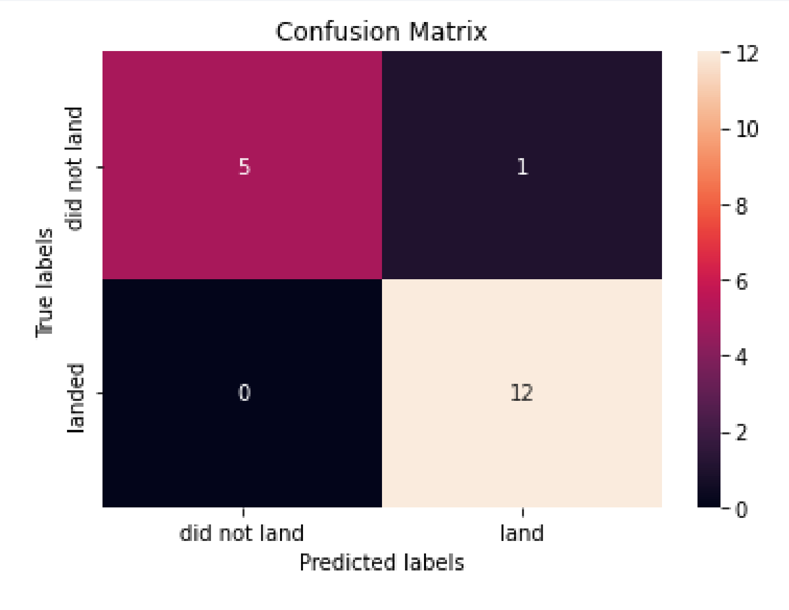
- **Logistic Regression** and **Support Vector Machine (SVM)** have the highest classification accuracy (both 0.944)

Confusion Matrix

- **Logistic Regression**



- **Support Vector Machine (SVM)**



Conclusions

- Orbits ES-L1, GEO, HEO and SSO have the highest success rates.
- **KSC LC-39A** had the most successful launches among all sites, but increasing payload weight seems to have negative impact on success.
- Success rates for SpaceX launches have been increasing over time and it seems they will reach the desired target soon.
- **Logistic Regression** and **Support Vector Machine (SVM)** have the highest classification accuracy and, therefore, deliver the best performance on test data.

Thank you!