



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Zhang Zhenhua  
2022.01.19



# Outline

---

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

# Executive Summary

---

We performed exploratory data analysis (EDA) and built a predictive model to understand and predict SpaceX Falcon 9 booster landing outcomes. Data was collected from SpaceX REST API and by webscraping the SpaceX Wikipedia page. EDA showed landing successes improved with number of flights and by year. The payload mass and launch orbit also appeared to play a role in landing outcomes. All variables were converted to onehot encoding format to build predictive models. Data was converted to numpy array and transformed by the standard scaler method in SciKitLearn. GridSearchCV method was used to search the best model parameters to predict Falcon 9 booster landings. Decision tree was the most accurate model for our prediction. The decision model incorporated landing pad, landing legs, booster version, flight number, and launch orbit as the most important variables for prediction.

# Introduction

---

In this capstone, we create a model to predict if the SpaceX 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. We explore a variety of predictive models to determine the best way to predict Falcon 9 booster landing.



Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Web scraping (source: Wikipedia) and REST API (source: SpaceX)
- Perform data wrangling
  - Data frame (using pandas to produce clean data of relevant features).
  - One hot encoding and normalization for predictive models
- 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 a numpy database which can be read by the SciKitLearn package
  - Search for best parameters for logistic regression, decision tree, support vector machine and K-nearest neighbor models and identify the best performing model for prediction



# Data Collection

---

- Data sets were collected SpaceX REST API and Wikipedia SpaceX html page. The datasets were transformed into a pandas data frame for further processing.

SpaceX  
REST API

Response  
JSON file

Parse JSON into  
Pandas data frame

HTML response  
From Wikipedia

Python Beautiful  
Soup object

Transform to Pandas  
data frame

# Data collection – SpaceX API

## 1. Get response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

## 2. Convert response to .json file

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

3. Apply custom functions to pull out data from .json
4. Assign values to dataframe

```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch\_dict.

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



# Data collection – Web scraping

## 4. Iterate through tables pull out relevant column data

### 1. Get response from HTML

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

```
response = requests.get(static_url).text
```

### 2. Convert HTML to a BeautifulSoup Object

```
soup = BeautifulSoup(response, 'html5lib')
```

### 3. Find table contents

```
html_tables = soup.find_all('table')
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant 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
```

### 5. Form a dataframe

```
In [112]: extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowhead
ers collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch
        h 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 dictionary
        if flag:
            extracted_row += 1
            ## Flight Number value
            ## TODO: Append the flight_number into launch_dict with key 'Flight
            No.'
            ## print(flight_number)
            datatimelist=date_time(row[0])
            launch_dict['Flight No.'].append(extracted_row)
            print("flight number:",extracted_row)

            ## Date value
            ## TODO: Append the date into launch_dict with key 'Date'
            date = datatimelist[0].strip(',')
```

# Data wrangling

- Key the landing outcomes as positive or not for each row in the dataset

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

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

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

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])  
bad_outcomes  
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

- Handle null values

```
In [39]: # Calculate the mean value of PayloadMass column  
x = data_falcon9['PayloadMass'].mean()  
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, x)  
# Replace the np.nan values with its mean value
```

```
In [40]: data_falcon9.isnull().sum()
```

```
Out[40]: 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

---

Launches with successful and failed landings were plotted to investigate relationships between various variables

- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass

# EDA with SQL

---

SQL queries were performed to investigate relationship of payload mass, launch site, launch date, booster versions, and landing success.

- Total payload mass
- Average payload mass
- First successful ground landing date
- Successful drone ship landing with payload between 4000 and 6000 kg
- Total number of successful and failed missions
- Boosters that carried the maximum payloads
- List the launch site and booster of 2015 failed landings
- Rank the failed landing outcome descriptions between 2010-06-04 and 2017-03-20

# Build an Interactive Map with Folium

---

- The location of the launch site were visualized in an interactive map to gain location specific insights about the launch sites used.
- Lat/Long coordinates were used to place markers of the launch sites in Folium.
- The successful and failed booster landings from the launch site was visualized using color coded circles.
- The launch site proximity to geographic feature and transport infrastructure was noted using distance lines.

# Build a Dashboard with Plotly Dash

---

- An interactive dashboard was created to quickly visualize relation ship between landing success and launch sites.
- A module was added to further show the relationship between payload and landing outcome at launch sites.



# Predictive Analysis (Classification)

1. All features were converted to one hot encoding

```
features_one_hot = pd.get_dummies(features[['Orbit', 'LaunchSite', 'LandingPad', 'Serial']])
features_one_hot.head()
```

3]:

	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	Orbit_LEO	Orbit_MEO	Orbit_PO	Orbit_SO	Orbit
0	0	0	0	0	0	1	0	0	0	
1	0	0	0	0	0	1	0	0	0	
2	0	0	0	0	1	0	0	0	0	
3	0	0	0	0	0	0	0	1	0	
4	0	0	1	0	0	0	0	0	0	

5 rows × 72 columns

2. Features were standardized to each other by the standard scaler method

```
In [6]: # students get this
transform = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
transform[0:5]
```

Out[6]: array([[ -1.71291154e+00, -1.94814463e-16, -6.53912840e-01,  
 -1.57589457e+00, -9.73440458e-01, -1.05999788e-01,  
 -1.05999788e-01, -6.54653671e-01, -1.05999788e-01,  
 -5.51677281e-01, 3.44342023e+00, -1.85695338e-01])

3. The data set was split to train and test sample sets to build and test the models

```
X_train, X_test, Y_train, Y_test = train_test_split(transform, Y, test_size=0.2, random_state=3)
```

4. The logistic regression, decision tree, k-nearest neighbors, and support vector machine were the models tested and scored

```
In [10]: parameters = {"C": [0.01, 0.1, 1], 'penalty': ['l2'], 'solver': ['lbfgs']}# l1 Lasso l2 ridge
lr=LogisticRegression()

In [11]: logreg_cv=GridSearchCV(lr, parameters)
logreg_cv.fit(X_train, Y_train)
```

# Results

---

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



Section 2

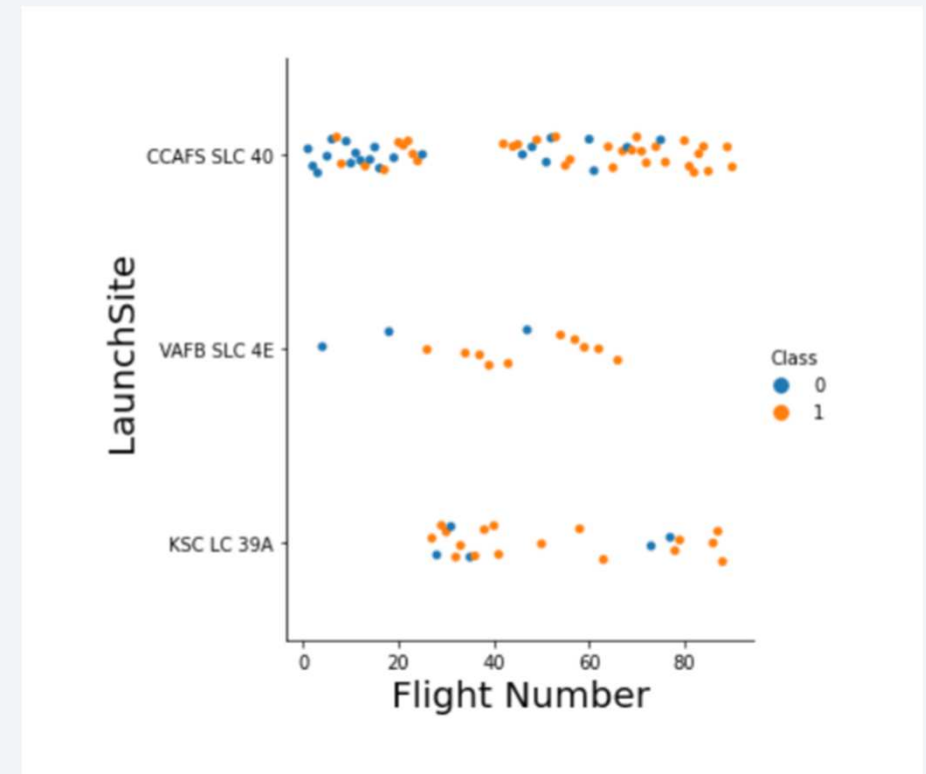
# Insights drawn from EDA



# Flight Number vs. Launch Site

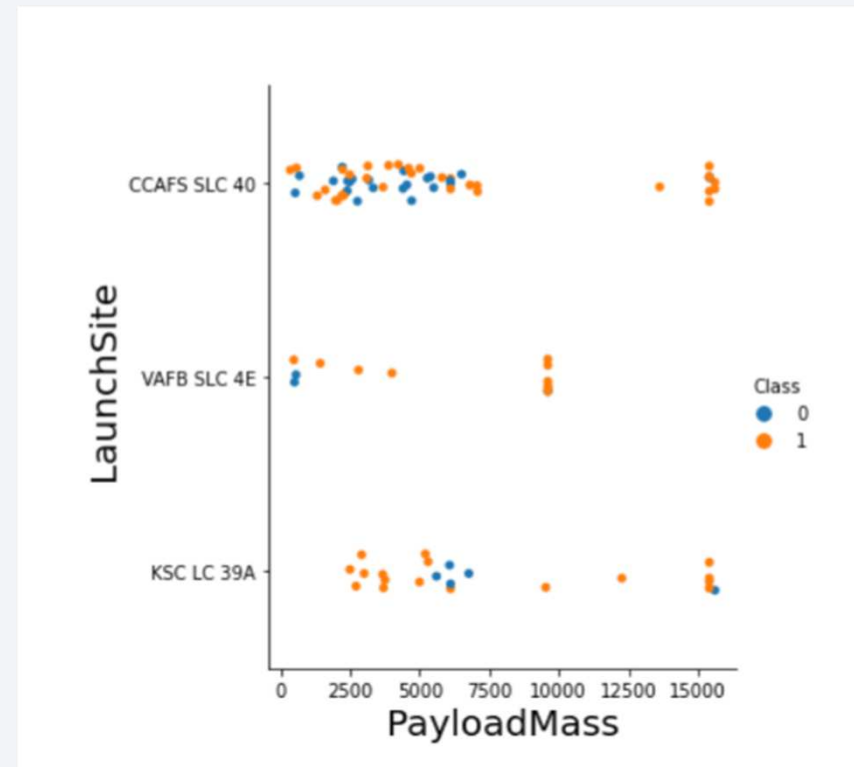
---

- Earlier flights occurred at CCAFS SLC 40 launch sites.
- Earlier flight numbers were more likely to have a negative landing outcome shown in blue (class = 0)
- In later flight numbers, launches occur from all site with majority of flights ending with successful landings shown in orange (class = 1)



# Payload vs. Launch Site

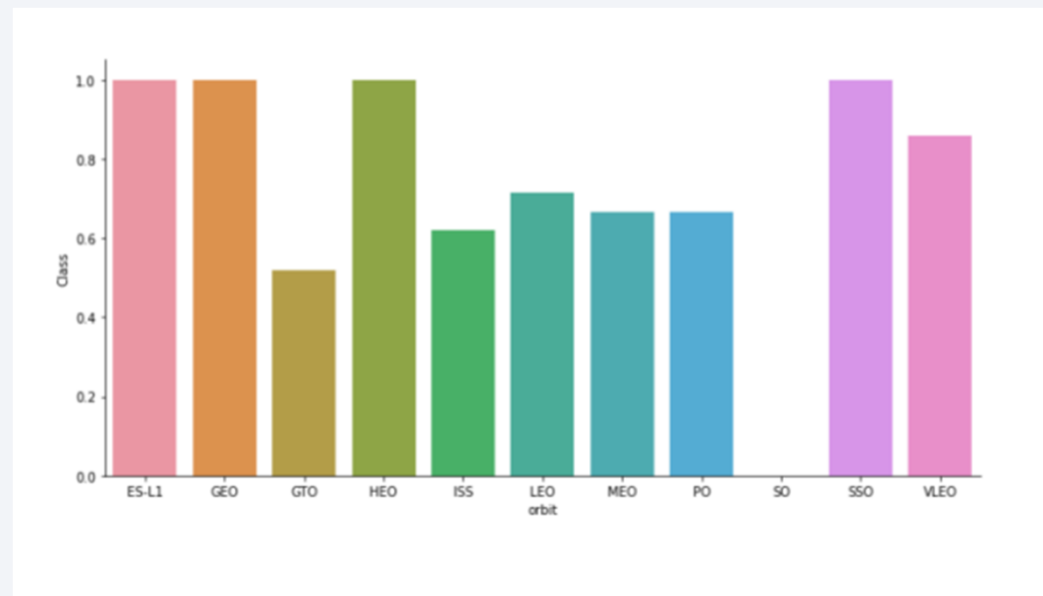
- Many of the lower payload mass launches occurred at CCAFS SLC 40.
- Higher payload mass launches appear to have higher landing success rate.



# Success Rate vs. Orbit Type

---

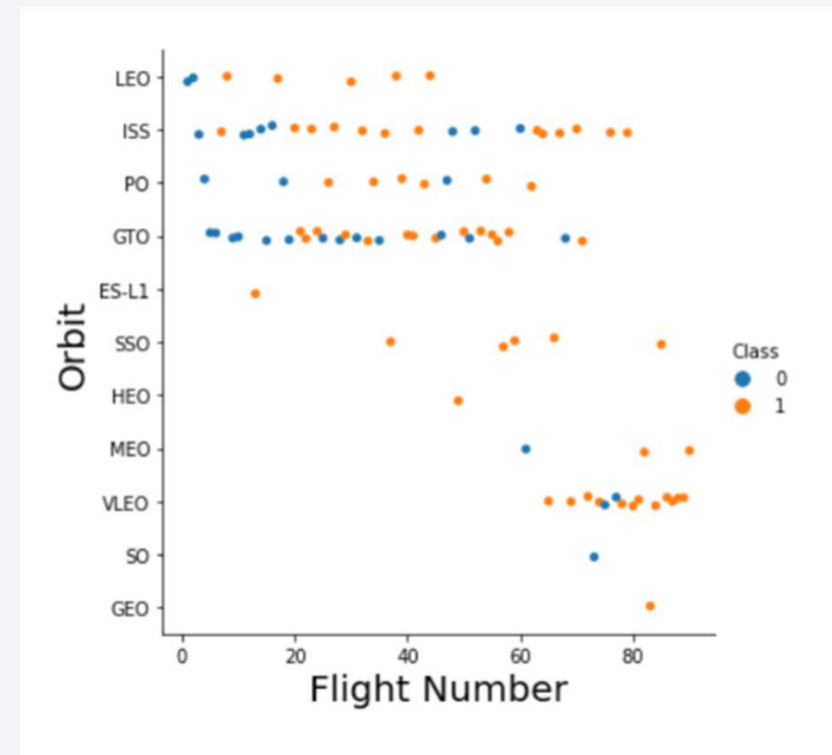
- Landing success rate differs by the orbit type. Some of the more difficult to reach orbit may require tight margins on fuel and weight. Its possible some orbits and weight combination make booster landings impossible or impractical.





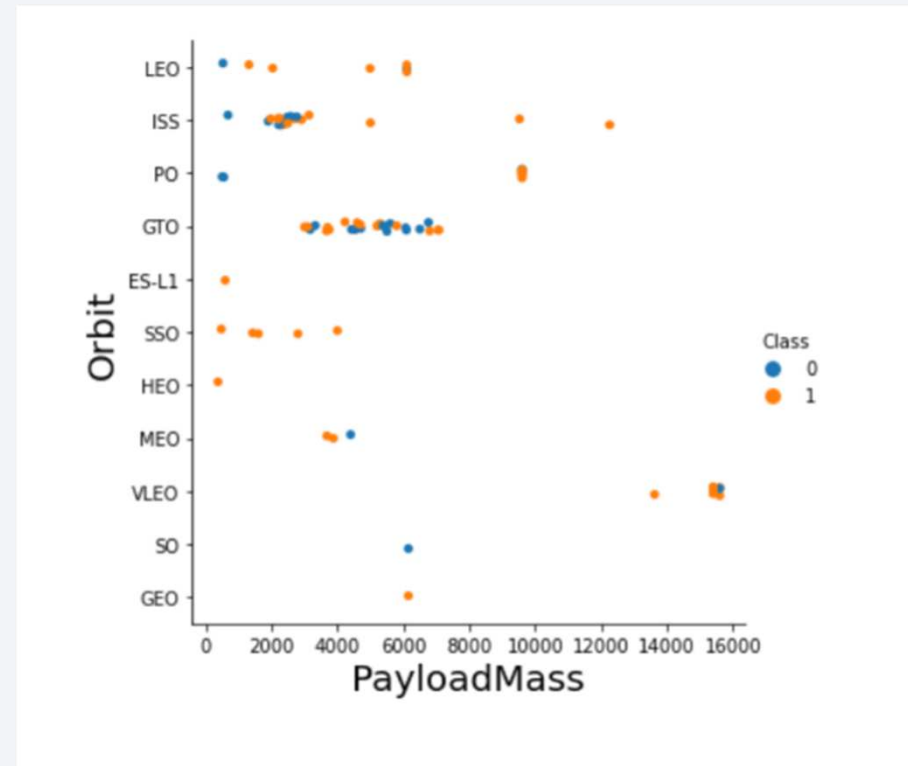
# Flight Number vs. Orbit Type

- Some orbits were not attempted in early flights
- Early flights had failed landing attempts
- LEO launches had successful landings early on. This trajectory might allow for more leeway for booster landings to occur because of the lower amount of energy required to reach this orbit.



# Payload vs. Orbit Type

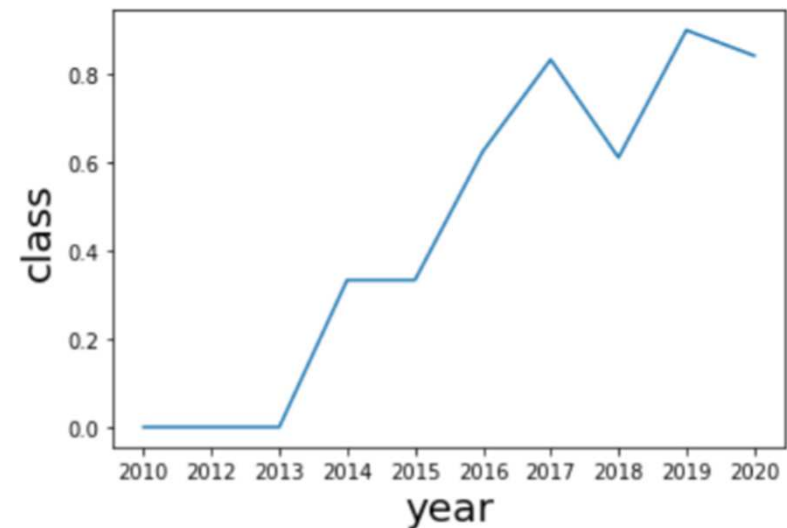
- Geostationary transfer (GTO) and International space station (ISS) orbits had the most failed landing outcomes.
- GTO and ISS orbits are higher energy and might require ocean landings. The payload mass were lower for these orbits also.
- Ocean landings might come with more risk.



# Launch Success Yearly Trend

---

- Yearly trend shows constant improvement in landing outcomes.
- One of the best predictors of landing outcomes may turn out to be flight number and year, because learnings from past landing attempts are incorporated in new flights.



# All Launch Site Names

---

'CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40'

```
: spacex_df['Launch Site'].unique()  
  
: array(['CCAFS LC-40', 'VAFB SLC-4E', 'KSC LC-39A', 'CCAFS SLC-40'],  
      dtype=object)
```

```
.0]: %sql select distinct launch_site from spacex  
      * ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a:  
Done.  
.0]: launch_site  
      CCAFS LC-40  
      CCAFS SLC-40  
      KSC LC-39A  
      VAFB SLC-4E
```

# Launch Site Names Begin with 'CCA'

```
] : %sql select *  
    from spacex  
    where launch_site like 'CCA%'  
    limit 5
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/blddb  
Done.
```

```
] : DATE time_utc booster_version launch_site payload payload_mass_kg orbit customer mission_outcome landing_outcome
```

2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

# Total Payload Mass

---

## Task 3

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

```
7]: %sql select sum(payload_mass_kg_) from spacex where customer like 'NASA%'
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
7]: 1
99980
```



# Average Payload Mass by F9 v1.1

---

## Task 4

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

```
: %sql select AVG(payload_mass__kg_) from spacex where booster_version like 'F9 v1.1%'
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
:      1
2534
```

# First Successful Ground Landing Date

---

## Task 5

List the date when the first successful landing outcome in ground pad was achieved.

*Hint: Use min function*

```
28]: %sql select MIN(DATE) from spacex where landing__outcome like '%ground pad%'
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

```
28]: 1
2015-12-22
```

## Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
[34]: %sql select distinct booster_version from spacex where (landing__outcome like '%drone ship%') AND (payload_mass__kg_ < 6000  
#landing__outcome like '%drone ship%'  
#AND (payload_mass__kg_ < 6000 AND payload_mass__kg_ > 4000)
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done.
```

```
[34]: booster_version
```

```
F9 FT B1021.2
```

```
F9 FT B1031.2
```

```
F9 FT B1020
```

```
F9 FT B1022
```

```
F9 FT B1026
```

# Total Number of Successful and Failure Mission Outcomes

---

List the total number of successful and failure mission outcomes

```
[52]: %sql SELECT mission_outcome, count(*) AS total FROM spacex GROUP BY mission_outcome
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

```
[52]:
```

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

# Boosters Carried Maximum Payload

---

```
%sql SELECT DISTINCT (booster_version) from spacex where payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) from spacex)
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done
```

**booster\_version**

F9 B5 B1048.4

F9 B5 B1048.5

F9 B5 B1049.4

F9 B5 B1049.5

F9 B5 B1049.7

F9 B5 B1051.3

F9 B5 B1051.4

F9 B5 B1051.6

F9 B5 B1056.4

F9 B5 B1058.3

F9 B5 B1060.2

F9 B5 B1060.3

# 2015 Launch Records

---

List the failed landing\_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
##%sql select * from spacex
```

```
%sql select booster_version, launch_site from spacex where landing__outcome = 'Failure (drone ship)' AND DATE like '2015%'
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

booster_version	launch_site
-----------------	-------------

F9 v1.1 B1012	CCAFS LC-40
---------------	-------------

F9 v1.1 B1015	CCAFS LC-40
---------------	-------------



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

```
79]: %%sql select * from spacex
```

```
%%sql SELECT * from (SELECT landing__outcome, count(*) AS total FROM spacex WHERE DATE between '2010-06-04' and '2017-03-20')
```

```
#DATE
```

```
%sql SELECT * FROM spacex where DAYNAME(DATE)='Friday' LIMIT 5
```

```
* ibm_db_sa://phq62734:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/blddb Done.
```

```
79]:
```

DATE	time_utc	booster_version	launch_site	payload	payload_mass_kg	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
2014-04-18	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
2016-03-04	23:35:00	F9 FT B1020	CCAFS LC-40	SES-9	5271	GTO	SES	Success	Failure (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)

Section 4

# Launch Sites Proximities Analysis



# Launchsite mapping with Folium

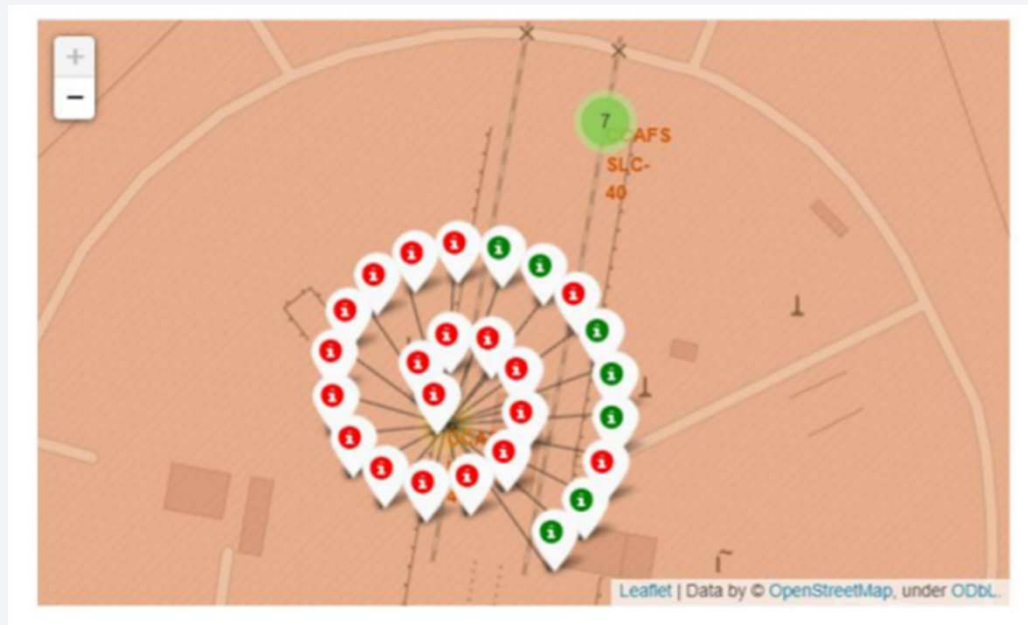
---



The launch sites are located on coasts

# Color-labeled launch records mapping

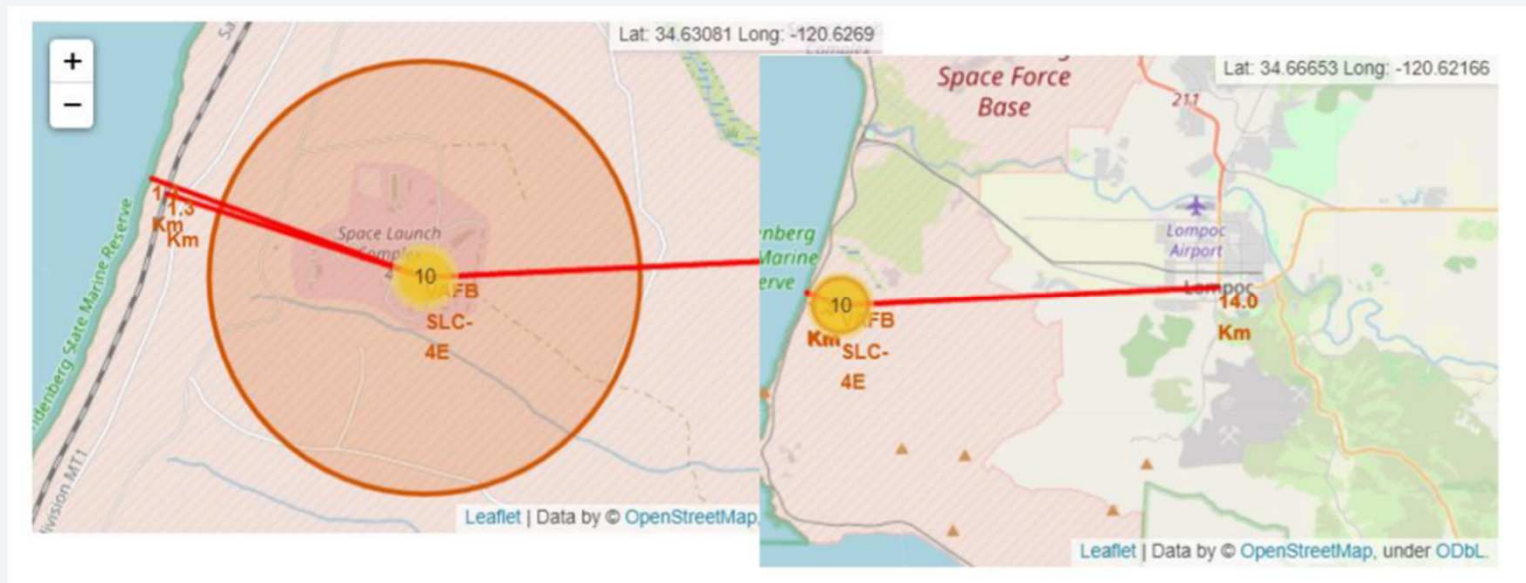
---



- The markers closer to the center occurred earlier. Many early launches did not end in a successful booster landing. Later launches are more likely to end in a landing.

# Launch site proximity to infrastructure

---



The launch site are located close to the ocean and close to rail and highway infrastructure.



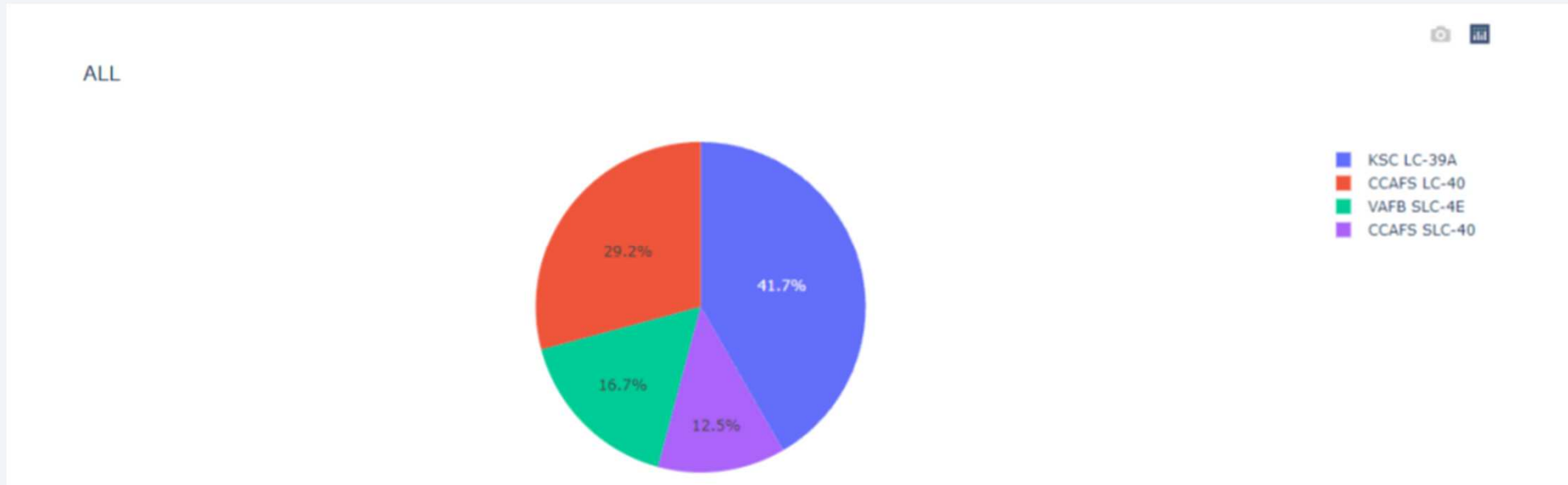


Section 5

# Build a Dashboard with Plotly Dash

# Landing successes from launch sites

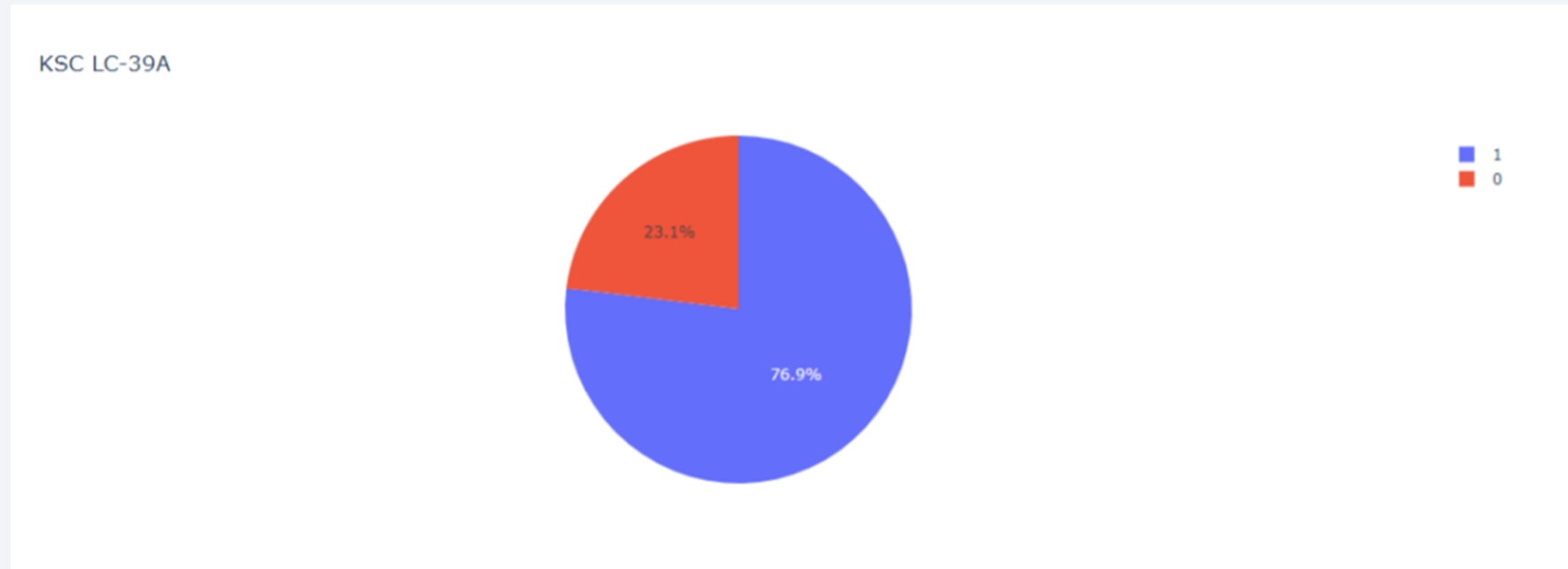
---



- Highest number of successful launch with booster landings occurred from KSC LC39A

## Launch site with highest landing success ratio

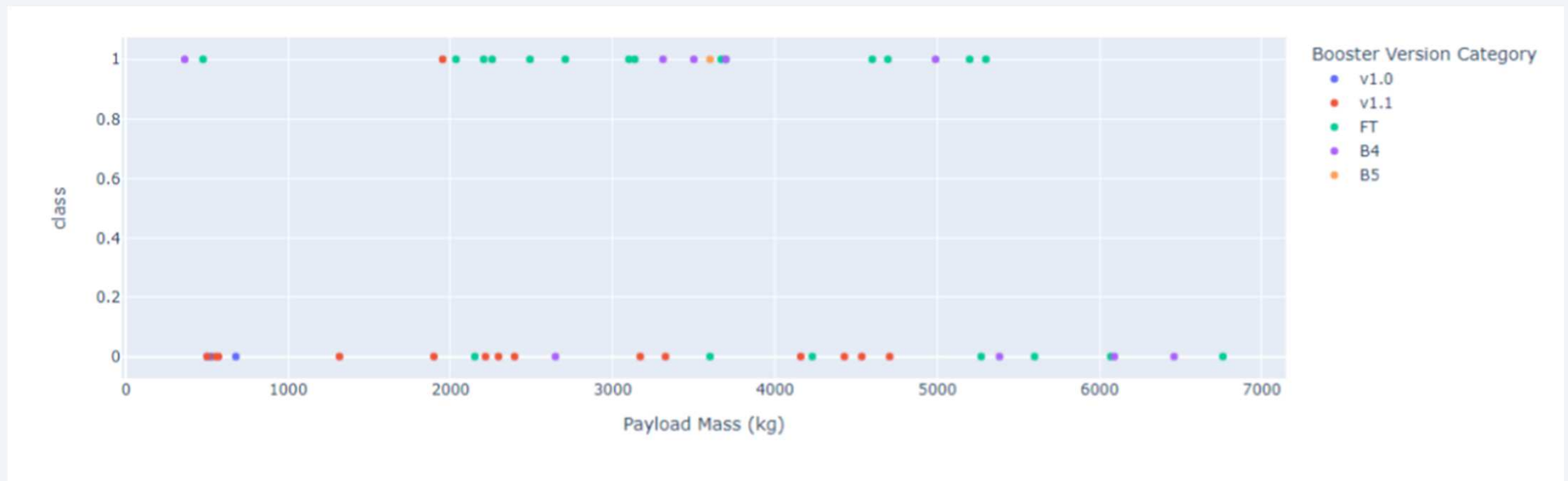
---



- From KSC LC-39A 76.9% of launches have ended in successful landings



# Payload vs. Launch Outcome



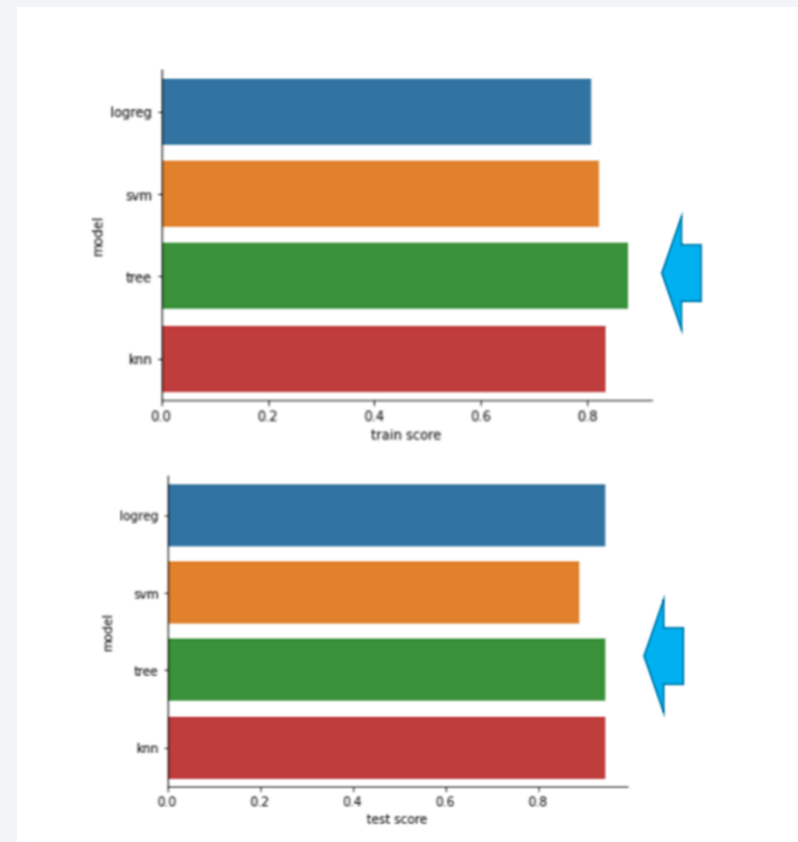
- When the payload mass is between 2000 to 4000 kg the success rate is higher than launches with payload size above 4000 kg. Payload size has some indication on landing outcome.

Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

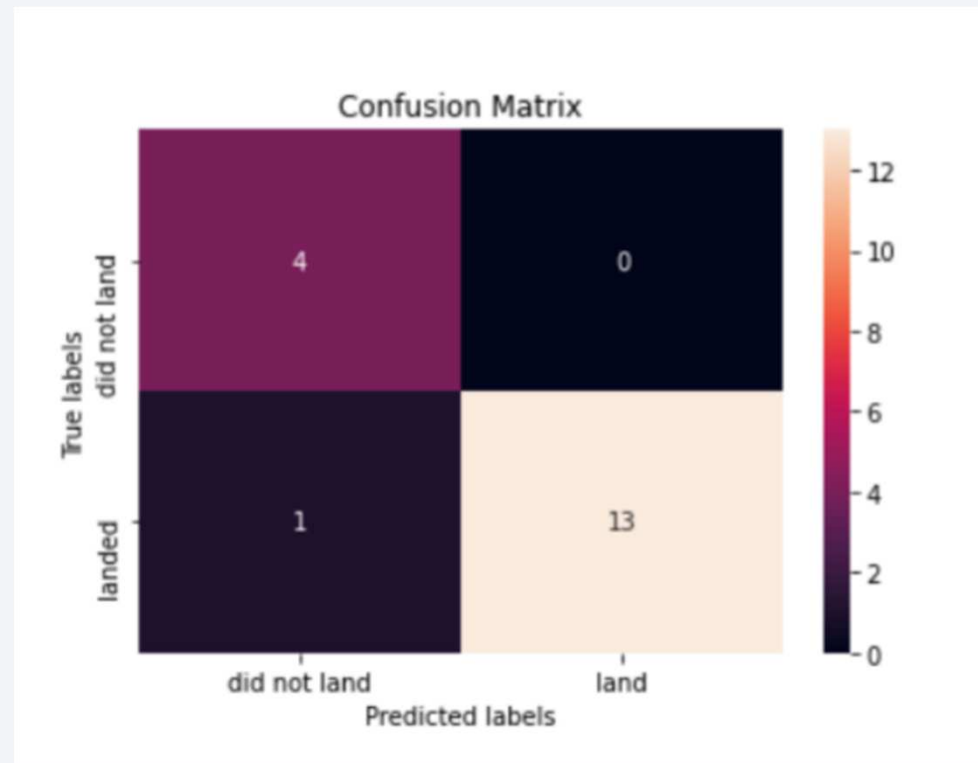
- The GridSearchCV tool was used to determine the best parameters for training 4 algorithms: Logistic regression (logreg), Support Vector Machine (SVM), Decision Tree (tree), and K -nearest neighbors (KNN)
- The model accuracy for the training set and test set was determined by scoring tool.
- Decision tree fit the training dataset and the test data set best. Although all algorithms performed similarly.



# Confusion Matrix

---

- Decision tree confusion matrix shows the model correctly predicted 13 landings and 4 failed landing outcomes. The model incorrectly labeled one landing as a failed landing outcome.



# Conclusions

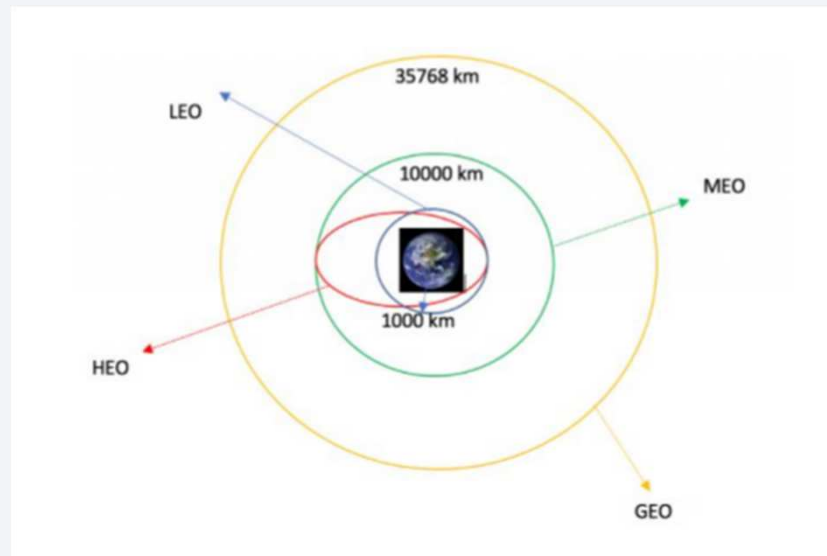
---

- Decision tree was determined to be the best predictive model for Falcon 9 booster landing on any given launch.
- EDA showed the number of flights, orbit type, launch site, and payload weight had an influence on landing outcome.
- Parameters in decision tree model incorporated some of the EDA findings, such as booster version which correlate with the flight number.
- Some of the weaknesses in the decision tree model is that it does not take into account many of the physical parameters might determine landing success such as payload mass or some of the orbit types.

# Appendix

---

- Depiction of orbit types. Some orbits require less flight distance than others



Thank you!

