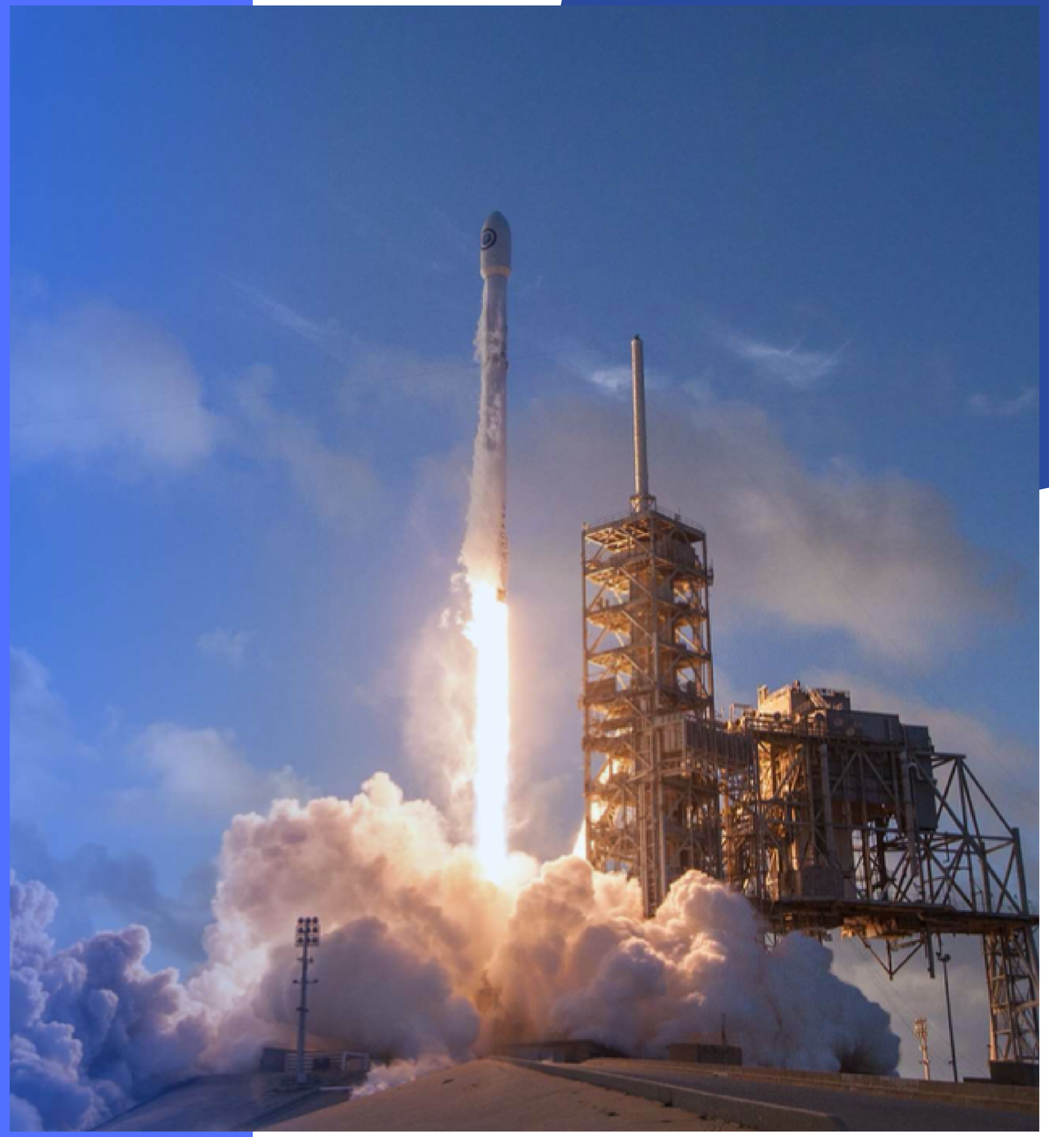
WINNING SPACE RACE

WITH DATA SCIENCE

HONG JIE WU

MARCH 10,2023

INDEX



EXECUTIVE SUMMARY

- SUMMARY OF METHODOLOGIES
  - - DATA COLLECTION THROUGH API
  - - DATA COLLECTION WITH WEB SCRAPING
  - - DATA WRANGLING
  - - EXPLORATORY DATA ANALYSIS (EDA) WITH SQL
  - - EXPLORATORY DATA ANALYSIS (EDA) WITH DATA VISUALIZATION
  - - INTERACTIVE VISUAL ANALYTICS WITH FOLIUM
  - - MACHINE LEARNING PREDICTION
  - • SUMMARY OF ALL RESULTS
  - - EXPLORATORY DATA ANALYSIS RESULT
  - - INTERACTIVE ANALYTICS IN SCREENSHOTS
  - - PREDICTIVE ANALYTICS RESULT

INTRODUCTION

Project background and context

SPACE X 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 SPACE X 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 SPACE X FOR A ROCKET LAUNCH. THIS GOAL OF THE PROJECT IS TO CREATE A MACHINE LEARNING PIPELINE TO PREDICT IF THE FIRST STAGE WILL LAND SUCCESSFULLY.

• Problems you want to find answers

- - WHAT FACTORS DETERMINE IF THE ROCKET WILL LAND SUCCESSFULLY?
- - THE INTERACTION AMONGST VARIOUS FEATURES THAT DETERMINE THE SUCCESS RATE OF A SUCCESSFUL LANDING.
- - WHAT OPERATING CONDITIONS NEEDS TO BE IN PLACE TO ENSURE A SUCCESSFUL LANDING PROGRAM.

1

METHODOLOGY

METHODOLOGY

EXECUTIVE SUMMARY

- DATA COLLECTION METHODOLOGY:
- DATA WAS COLLECTED USING SPACEX API AND WEB SCRAPING FROM WIKIPEDIA.
- PERFORM DATA WRANGLING
- ONE-HOT ENCODING WAS APPLIED TO CATEGORICAL FEATURES
- PERFORM EXPLORATORY DATA ANALYSIS (EDA) USING VISUALIZATION AND SQL
- PERFORM INTERACTIVE VISUAL ANALYTICS USING FOLIUM AND PLOTLY DASH
- PERFORM PREDICTIVE ANALYSIS USING CLASSIFICATION MODELS
- HOW TO BUILD, TUNE, EVALUATE CLASSIFICATION MODELS

DATA COLLECTION

THE DATA WAS COLLECTED USING VARIOUS METHODS

- - DATA COLLECTION WAS DONE USING GET REQUEST TO THE SPACEX API.
- - NEXT, WE DECODED THE RESPONSE CONTENT AS A JSON USING .JSON() FUNCTION CALL AND TURN IT INTO A PANDAS DATAFRAME USING .JSON_NORMALIZE().
- - WE THEN CLEANED THE DATA, CHECKED FOR MISSING VALUES AND FILL IN MISSING VALUES WHERE NECESSARY.
- - IN ADDITION, WE PERFORMED WEB SCRAPING FROM WIKIPEDIA FOR FALCON 9 LAUNCH RECORDS WITH BEAUTIFULSOUP.
- - THE OBJECTIVE WAS TO EXTRACT THE LAUNCH RECORDS AS HTML TABLE, PARSE THE TABLE AND CONVERT IT TO A PANDAS DATAFRAME FOR FUTURE ANALYSIS.

DATA COLLECTION – SPACEX AP

1. Get request for rocket launch data using API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

2. Used json_normalize method to convert json result to dataframe

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successful with the 200 status response code

```
In [10]: response.status_code
Out[10]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [14]: # Use json_normalize meethod to convert the json result into a dataframe
         json = requests.get(static_json_url).json()
         df2 = pd.json_normalize(json)
         df2.head()
         df = pd.read_json(static_json_url)
```

3. Extract only useful columns and store in lists

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
In [16]: # Storing a copy of 'df' dataframe in 'data'
         data = df

         # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
         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 rocket boosters and rows that have multiple payloads in a sing
         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 list and replace the feature.
         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 leaving the time
         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)]
```

4. Construct dataset by combining columns into a dictionary and create a Pandas data frame

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
In [26]: 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.

```
In [27]: # Create a data from launch_dict
         data_falcon9 = pd.DataFrame(launch_dict)
```

5. Filter for only Falcon 9 launches and reset launch number

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
In [29]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
```

Now that we have removed some values we should reset the FlightNumber column

```
In [30]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
         data_falcon9
```

Out[30]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Lc
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-9

DATA COLLECTION – SCRAPING

1. Request Falcon 9 Launch page from URL

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
In [10]: # use requests.get() method with the provided static_url
         # assign the response to a object
         data = requests.get(static_url)
         data.status_code
Out[10]: 200
```

2. Create BeautifulSoup object from HTML

Create a `BeautifulSoup` object from the HTML `response`

```
In [12]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
         soup = BeautifulSoup(data.text, 'html.parser')
```

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

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
In [34]: # 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')
```

4. Create a data frame by parsing the launch HTML tables and export to csv file

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
In [38]: 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
         launch_dict['Version Booster']=[]
         launch_dict['Booster landing']=[]
         launch_dict['Date']=[]
         launch_dict['Time']=[]
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
In [20]: df=pd.DataFrame(launch_dict)
```

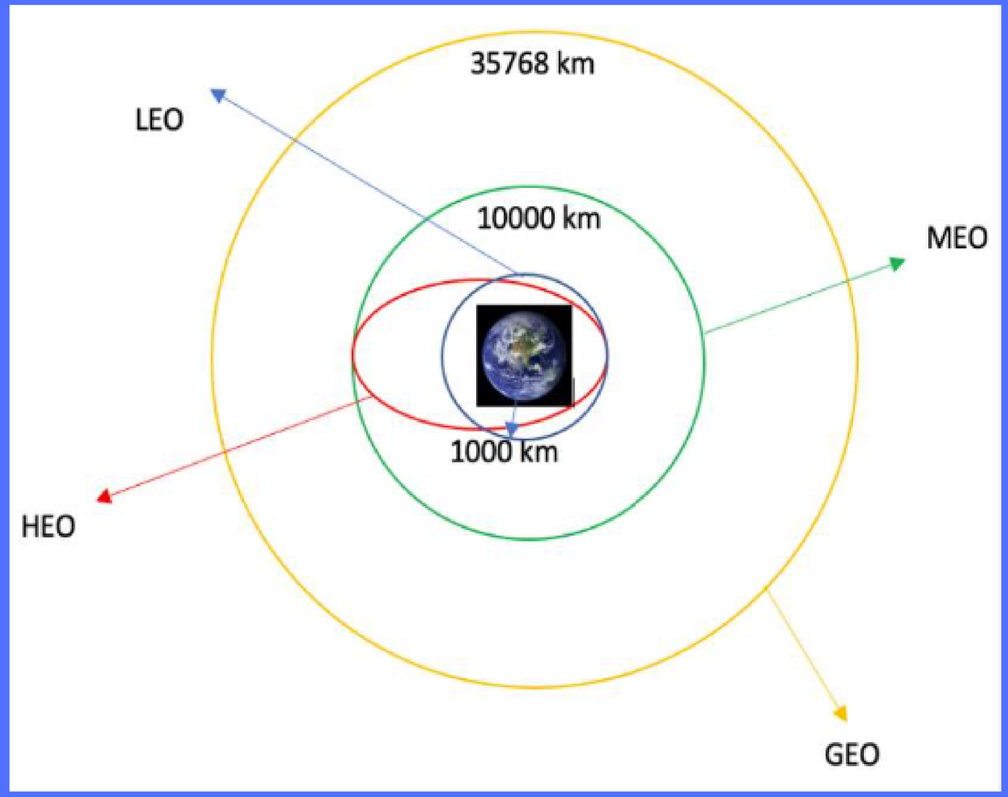
We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

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

```
In [21]: df.to_csv('spacex_web_scraped.csv', index=False)
```

DATA WRANGLING



- PERFORMED EXPLORATORY DATA ANALYSIS AND DETERMINED THE TRAINING LABELS.
- CALCULATED THE NUMBER OF LAUNCHES AT EACH SITE, AND THE NUMBER AND OCCURRENCE OF EACH ORBITS
- CREATED LANDING OUTCOME LABEL FROM OUTCOME COLUMN AND EXPORTED THE RESULTS TO CSV.

EDA WITH DATA VISUALIZATION

EXPLORED THE DATA BY VISUALIZING THE RELATIONSHIP BETWEEN THE FOLLOWING:

- FLIGHT NUMBER AND LAUNCH SITE,
- PAYLOAD AND LAUNCH SITE,
- SUCCESS RATE OF EACH ORBIT TYPE,
- FLIGHT NUMBER AND ORBIT TYPE,
- THE LAUNCH SUCCESS YEARLY TREND.

BUILD AN INTERACTIVE MAP WITH FOLIUM

- MARKED ALL LAUNCH SITES, AND ADDED MAP OBJECTS SUCH AS MARKERS, CIRCLES, LINES TO INDICATE THE SUCCESS OR FAILURE OF LAUNCHES FOR EACH SITE ON THE FOLIUM MAP.
- ASSIGNED THE FEATURE LAUNCH OUTCOMES (FAILURE OR SUCCESS) TO CLASS 0 AND 0 FOR FAILURE, AND 1 FOR SUCCESS.
- CHANGED THE COLOUR OF THE MARKER TO INDICATE SUCCESS (GREEN) OR FAILURE (RED)
- CALCULATED THE DISTANCES BETWEEN A LAUNCH SITE TO ITS PROXIMITIES AND CREATED LINE OBJECTS FOR A VISUAL REPRESENTATION
- ANSWERED THE FOLLOWING QUESTIONS FOR INSTANCE:

- ARE LAUNCH SITES NEAR RAILWAYS, HIGHWAYS AND COASTLINES. – NO, NO, YES

- DO LAUNCH SITES KEEP CERTAIN DISTANCE AWAY FROM CITIES. - YES

BUILD A DASHBOARD WITH PLOTLY DASH

- PLOTLY DASH IS AN INTERACTIVE DASHBOARD
- THE DASHBOARD CONTAINS TWO CHARTS
- PIE CHARTS SHOWING THE TOTAL LAUNCHES BY A CERTAIN SITES. THIS CHART IS USEFUL AS YOU CAN VISUALIZE THE DISTRIBUTION OF LANDING OUTCOMES ACROSS ALL LAUNCH SITES OR SHOW THE SUCCESS RATE OF LAUNCHES ON INDIVIDUAL SITES.
- SCATTER GRAPH SHOWING THE RELATIONSHIP WITH OUTCOME AND PAYLOAD MASS (KG) FOR THE DIFFERENT BOOSTER VERSION. THIS CHART IS USEFUL AS YOU CAN VISUALIZE HOW DIFFERENT VARIABLES AFFECT THE LANDING OUTCOMES

PREDICTIVE ANALYSIS (CLASSIFICATION)

- COMPLETED THE FOLLOWING STEPS
  - LOADED THE DATA USING NUMPY AND PANDAS
  - STANDARDIZED AND TRANSFORMED THE DATA
  - SPLIT OUR DATA INTO TRAINING AND TESTING.
  - • BUILT DIFFERENT MACHINE LEARNING MODELS AND TUNE DIFFERENT HYPERPARAMETERS (SVM, DECISION TREES, K-NEAREST NEIGHBOURS AND LOGISTIC REGRESSION) USING GRIDSEARCHCV.
  - • USED TEST DATA TO EVALUATE MODELS BASED ON THEIR ACCURACY SCORES AND CONFUSION MATRIX
- FOUND THE BEST PERFORMING CLASSIFICATION MODEL

RESULTS

- EXPLORATORY DATA ANALYSIS INDICATED THE SUCCESS RATE OF THE FALCON 9 LANDINGS WAS 66%
- INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS IN THE FOLLOWING SLIDES
- PREDICTIVE ANALYSIS RESULTS INDICATED THE DECISION TREE ALGORITHM WAS THE BEST AT 87% ACCURACY

2

INSIGHTS DRAWN FROM EDA

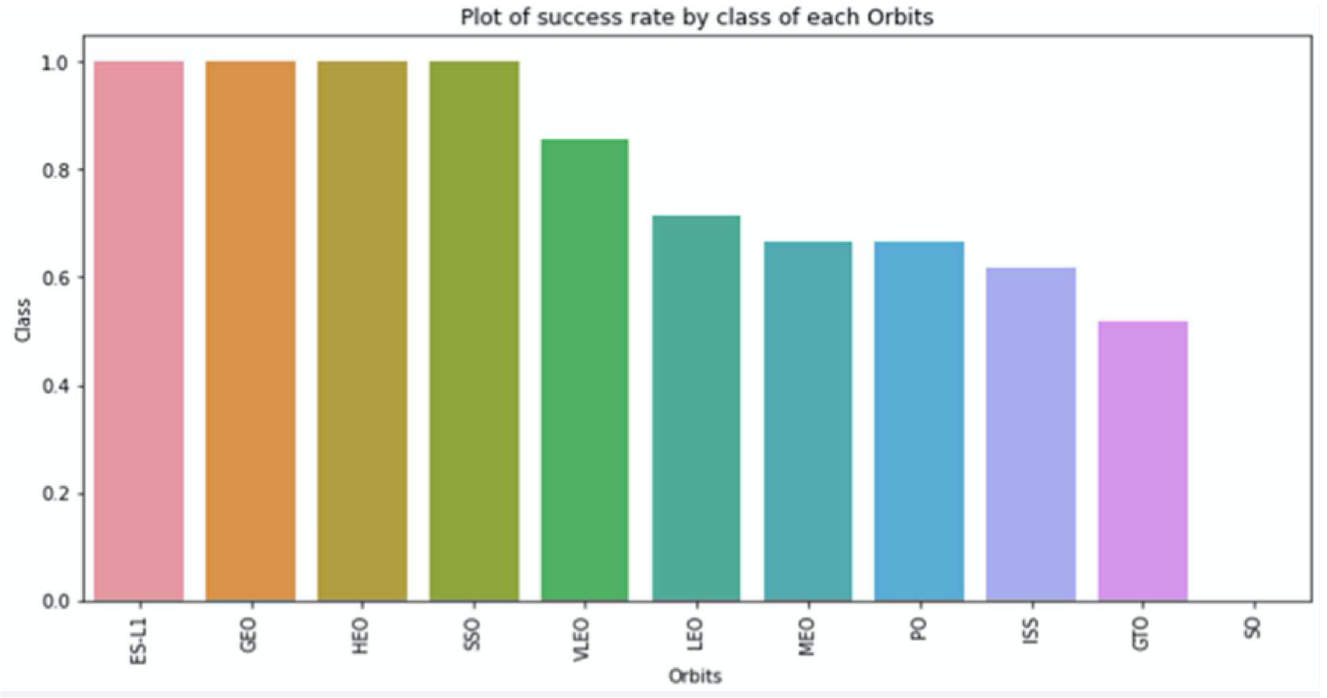
FLIGHT NUMBER VS. LAUNCH SITE

- THE MORE FLIGHTS AT A LAUNCH SITE, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- THE GREATER THE PAYLOAD MASS AT CCAFS SLC40, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- NO ROCKETS LAUNCHED FOR HEAVYPAYLOAD MASS (GREATER THAN 10000) AT VAFB-SLC

FLIGHT NUMBER VS. LAUNCH SITE

THE FOLLOWING ALL HAD THE HIGHEST SUCCESS RATE:

- ES-L1 • GEO
- HEO
- SSO
- VLEO



3

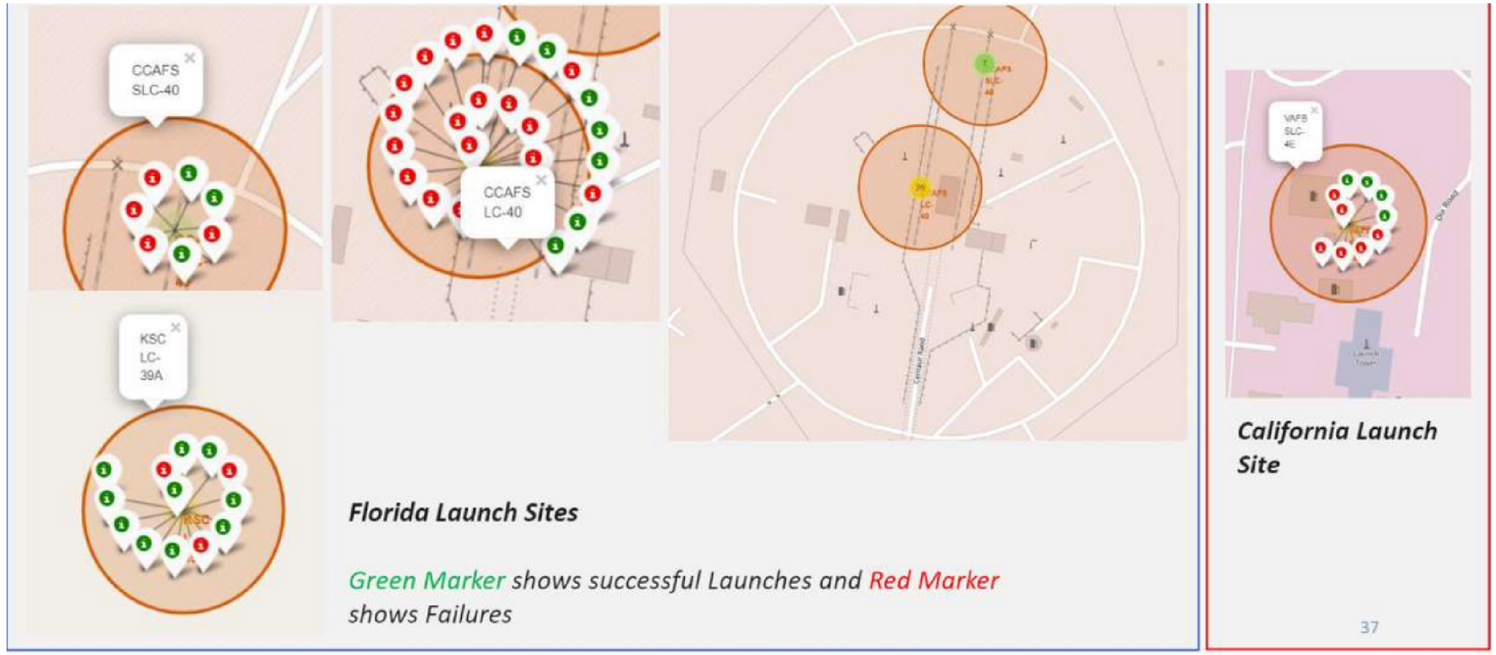
LAUNCH SITES PROXIMITIES ANALYSIS

SPACEX LAUNCH SITES LOCATIONS

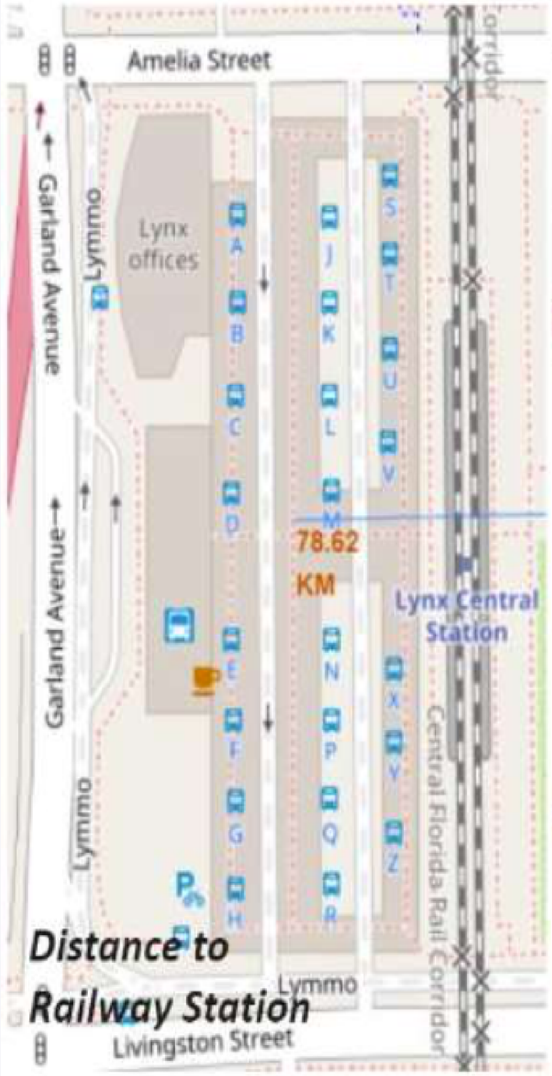
- THE YELLOW MARKERS ARE INDICATORS OF THE CLUSTER OF LOCATIONS OF THE SPACEX LAUNCH SITES
- ALL LAUNCH SITES ARE SITUATED IN THE US.
- THE LAUNCH SITES HAVE BEEN
- STRATEGICALLY PLACED NEAR THE COAST

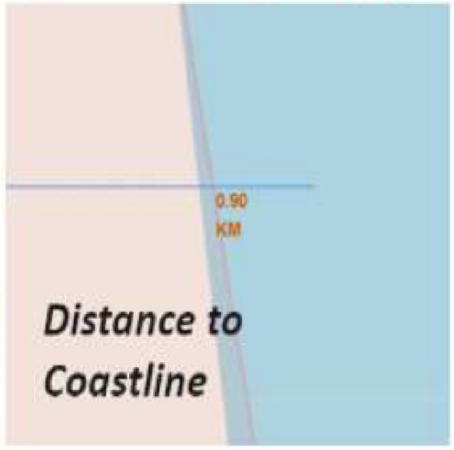
LAUNCH SITES WITH COLOUR LABELS

- THE MORE FLIGHTS AT A LAUNCH SITE, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- THE GREATER THE PAYLOAD MASS AT CCAFS SLC40, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- NO ROCKETS LAUNCHED FOR HEAVYPAYLOAD MASS (GREATER THAN 10000) AT VAFB-SLC



LAUNCH PROXIMITY TO LANDMARKS





- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

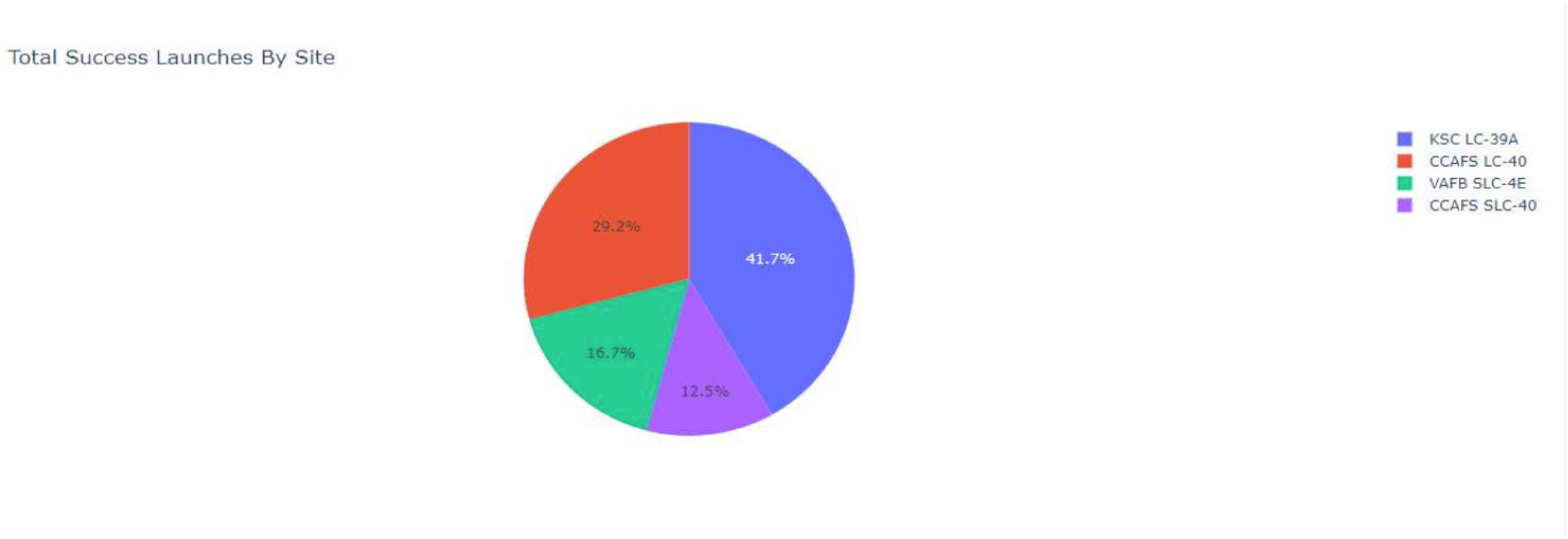
FLIGHT NUMBER VS. LAUNCH SITE

- THE MORE FLIGHTS AT A LAUNCH SITE, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- THE GREATER THE PAYLOAD MASS AT CCAFS SLC40, THE GREATER THE SUCCESS RATE AT A LAUNCH SITE.
- NO ROCKETS LAUNCHED FOR HEAVYPAYLOAD MASS (GREATER THAN 10000) AT VAFB-SLC

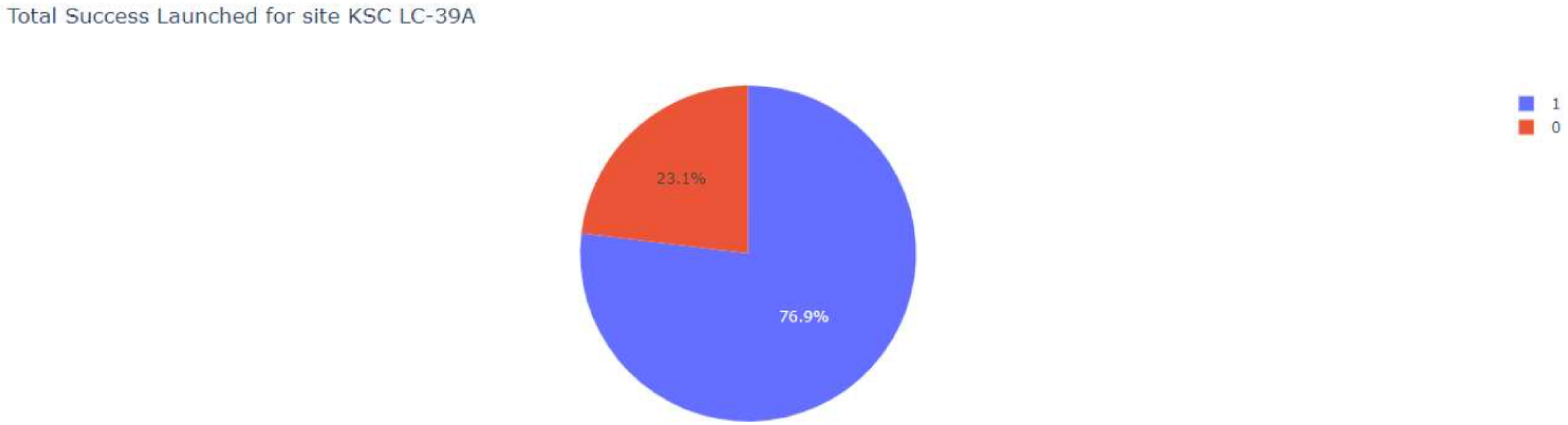
4

BUILD A BASHBOARD WITH PLOT DASH

TOTAL SUCCESSFUL LAUNCHES BY SITE



LAUNCH SITE WITH HIGHEST SUCCESS RATIO



5

PREDICTIVE ANALYSIS CLASSIFICATION

CLASSIFICATION ACCURACY

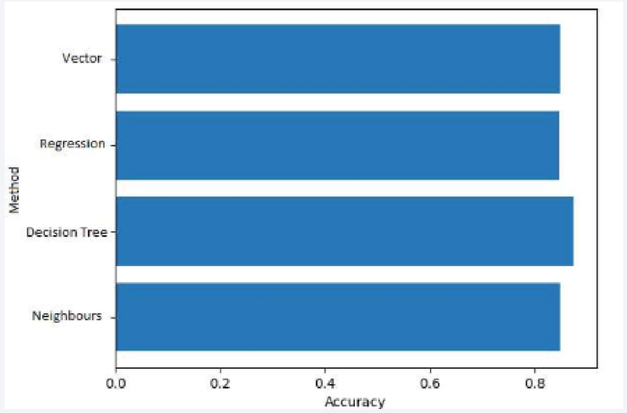
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

```
import numpy as np
import matplotlib.pyplot as plt

plt.barh(['Neighbours', 'Decision Tree', 'Regression', 'Vector'],[knn_cv.best_score_,tree_cv.best_score_,logreg_cv.best_score_,svm_cv.best_score_])
plt.xlabel('Accuracy')
plt.ylabel('Method')
plt.show()
```



The Decision Tree algorithm was the best at 87% accuracy

CONFUSION MATRIX

- THE CONFUSION MATRIX FOR THE BEST PERFORMING DECISION TREE CLASSIFIER SHOWS THAT THE CLASSIFIER CAN DISTINGUISH BETWEEN THE DIFFERENT CLASSES.
- THE MAIN PROBLEM ARE FALSE POSITIVES
- UNSUCCESSFUL LANDING MARKED AS SUCCESSFUL LANDING BY THE CLASSIFIER.

CONCLUSIONS

1. The more flights at a launch site, the greater the success rate.
2. Launch success rate steadily increased from 2013 to 2020.
3. Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
4. KSC LC-39A had the most successful launches of any sites.
5. The Decision Tree classifier is the best machine learning algorithm for this task.

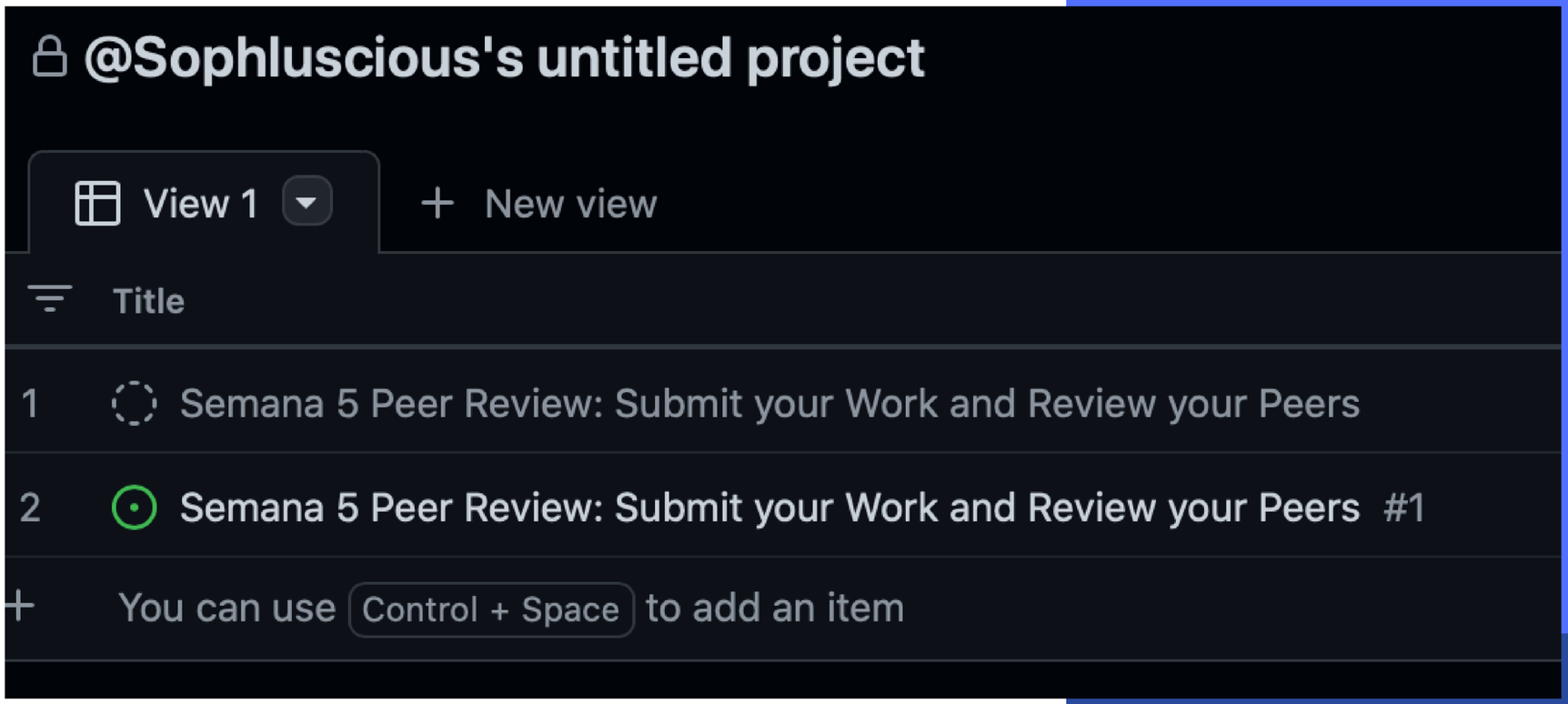
APENDIX

1

2

3

4



THANK YOU !