Winning Space Race with data Science

Roman Ondik
3-10-2021

Outline

Executive Summary

- Summary of methodologies
  - Collection of data was done using APIs,SQL, and WebScraping methods
  - Data wrangling and analysis
  - Maps and interactive maps using Folium
  - Predictive analysis for models
- Summary of all results
  - Interactive visual analytics with Folium
  - Machine learning prediction models

Introduction

Project background and context

- We want to predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Problems you want to find answers

- What are the conditions for a successful landing.
- How was the outcome dependent on different variables and what were they.

Methodology

[Link to GitHub Repository]

Methodology

Executive Summary

Data collection methodology:

- Method 1: using SpaceX REST API
- Method 2: WebScraping Wikipedia

Perform data wrangling

- Data was cleaned from unneccessary columns and converted using one hot encoding

Perform exploratory data analysis (EDA) using visualization and SQL

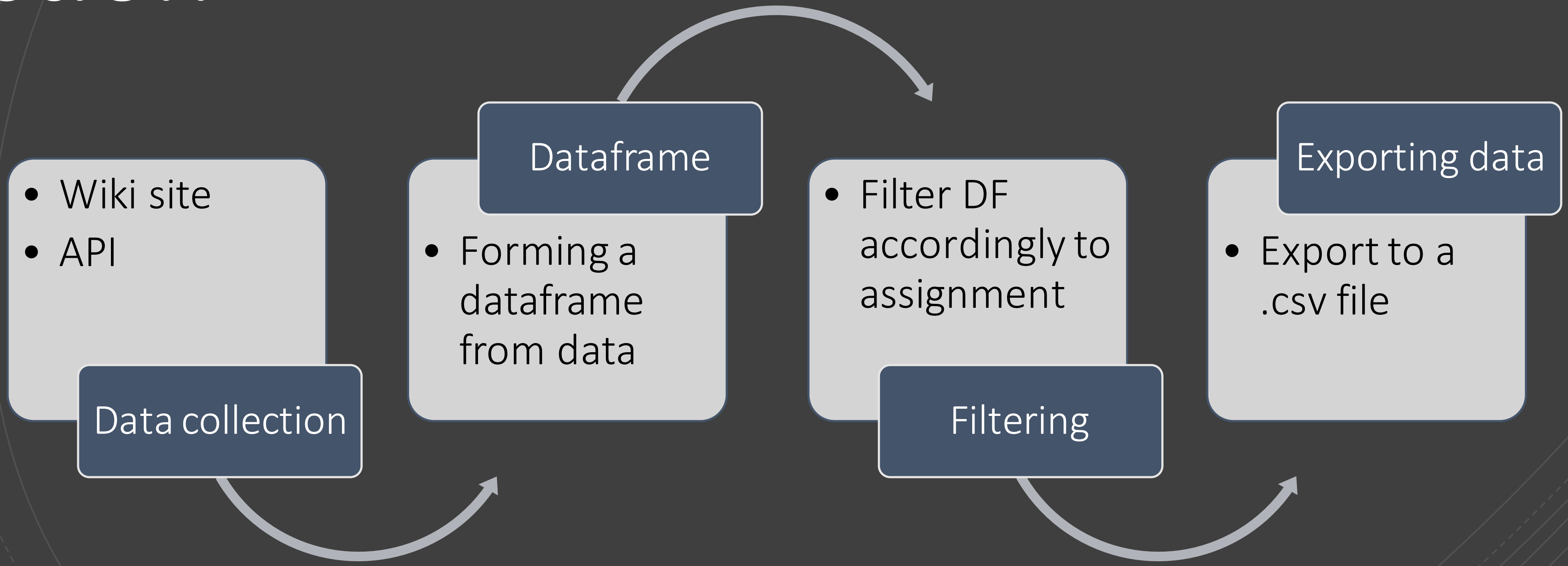
Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

- How to build, tune, evaluate classification models

Data Collection

- Collection of data is a technique where several methods can be used to gather data from an external source such as an API or a Website. In this project we used SpaceX REST API and Web Scraping from a Wikipedia page.



link

Data Collection – SpaceX API

Get response from API

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

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

Convert to .json

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

Clean data via functions

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

Assign list to dictionary

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
```

Filter data and export to .csv

```
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

link

Data Collection - Scraping

Get response from html

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

Create beautiful soup object

```
soup = BeautifulSoup(web.content, "html.parser")
```

Find tables in web

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

Get column names

```
column_names = []
tabl = soup.find_all('th')
for x in range(len(tabl)):
        try:
            name= extract_column_from_header(tabl[x])
            if (name is not None and len(name) > 0):
                column_names.append(name)
        except:
                pass
```

Creating a dictionary and appending data to keys

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']
launch_dict['Flight No.'] = []
```

Converting dictionary to dataframe

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

Converting dataframe to csv

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

link

Data Wrangling

Calculate the number of launches from each site

```
df['LaunchSite'].value_counts()
```

Calculate number and occurence of each orbit

```
df['Orbit'].value_counts()
```

Calculate the number of mission outcomes per orbit type

```
landing_outcomes= df['Outcome'].value_counts()
landing_outcomes
```

Create a landing outcome label from"Outcome" column

```
df['Class']=landing_class
```

Export to .csv

```
df.to_csv("dataset_part\_2.csv", index=False)
```

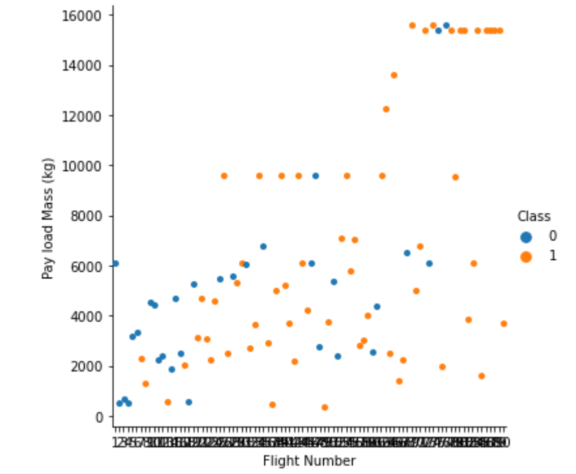
- Data wrangling is the process of transforming and mapping data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes such as analytics. The goal of data wrangling is to assure quality and useful data. Data analysts typically spend the majority of their time in the process of data wrangling compared to the actual analysis of the data.

link

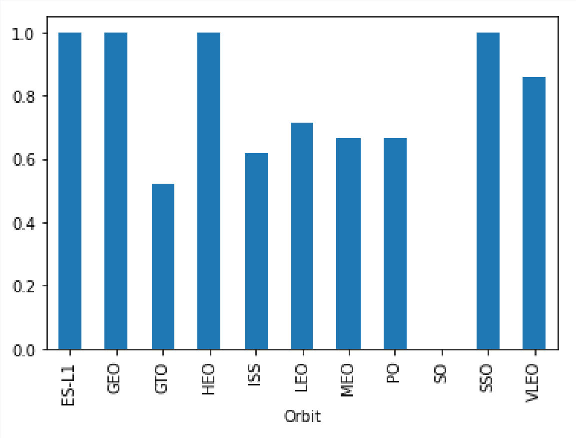
EDA with Data Visualization

- Payload and Flight#
- Launch Site and Flight#
- Launch Site and Payload
- Orbit type and Flight#
- Payload and Orbit type

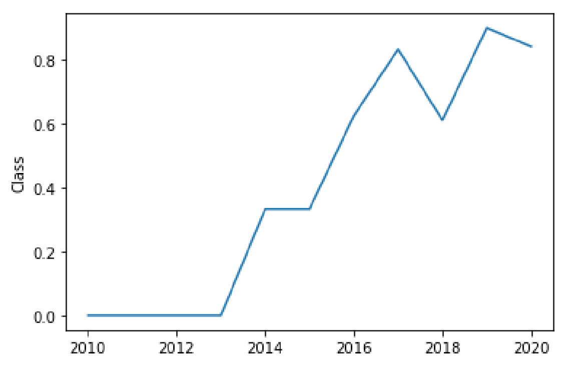
Scatter plots are used to observe relationships between variables.



Bar graph is a graph that shows rectangles of different heights, which depend on the value that each category has.



A line graph is showing the continuous change of variables. It has values connected with a line.



link

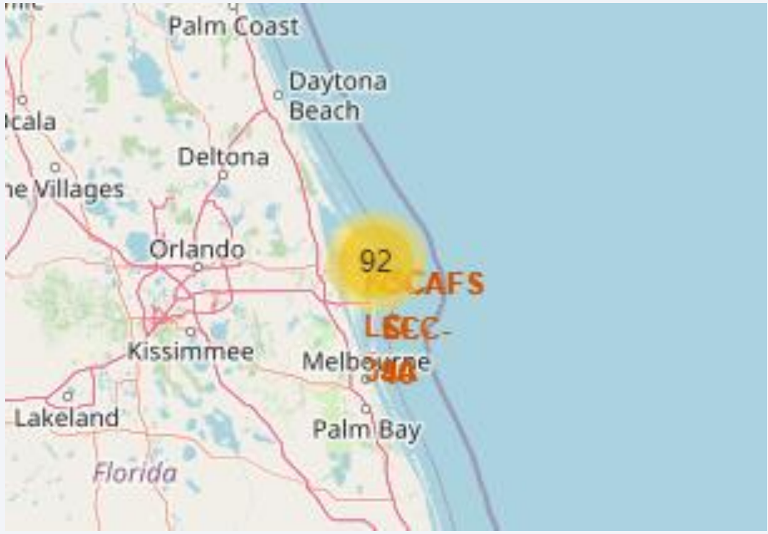
EDA with SQL

- It is the most used language for Relational Databases. It is designed to "query" information from databases.
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship 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. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

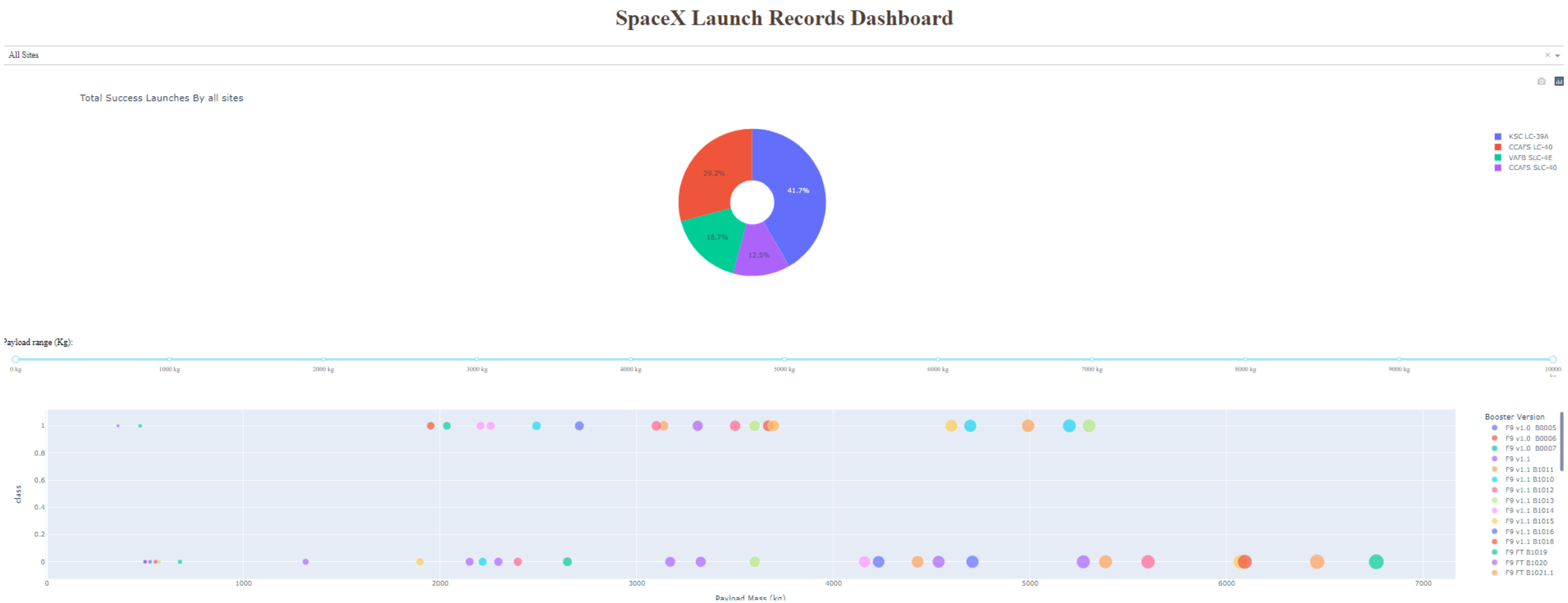
[link]

Build an Interactive Map with Folium

- Maps are a useful tool to show the location/distance/geographical and numeric occurence of data.
- In this part we used the coordinates of the launch sites (longitude, latitude) and placed a marker with the name of the launch site on them along with a circle around the marker. Then we used "Class" column to determine the color of the marker (red had value of 0 and green had 1).



folium.Marker	folium.Circle	folium.Icon	Folium.PolyLine	MarkerCluster()
• To make a marker on the map	• Create a circle around a marker	• To have an icon on the map	• To have a line=distance displayed on the map	• To have multiple markers in one spot



Build a Dashboard with Plotly Dash

- **Components of the dashboard:**
- Pie chart which displays the success rates of launch sites, or can be drilled down using a dropdown to display only the success rate of selected site.
- Scatter graph which is showing the success rates per payload mass for each booster version

[link]

Predictive Analysis (Classification)

- Load data into df
- Transform data into NumPy Arrays
- Standardize data
- Split into train/test sets
- Check the number of test samples
- Decide on ML algorithms to use
- Set parameters and algorithm to GridSearchCV
- Fit the datasets into GridSearch objects and train the dataset

Evaluating model

- Check accuracy for each model
- Get best hyperparameters
- Plot confusion matrix

↓

Improving model

- Engineering the features
- Tuning the algorithm

↓

Finding the best performing model

- Evaluate which model has the best accuracy score and choose that one

link

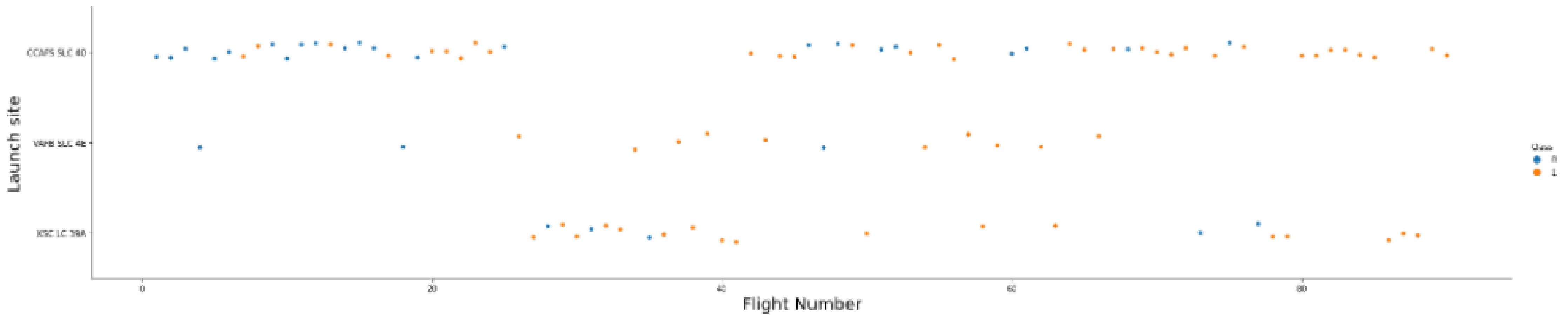
Results

Exploratory data analysis results

Interactive analytics demo in screenshots

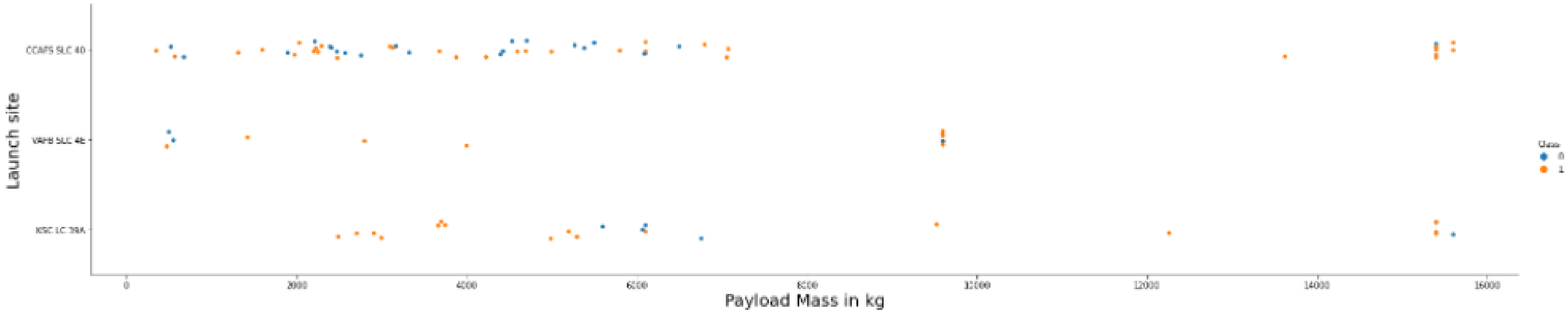
Predictive analysis results

Insights drawn from EDA



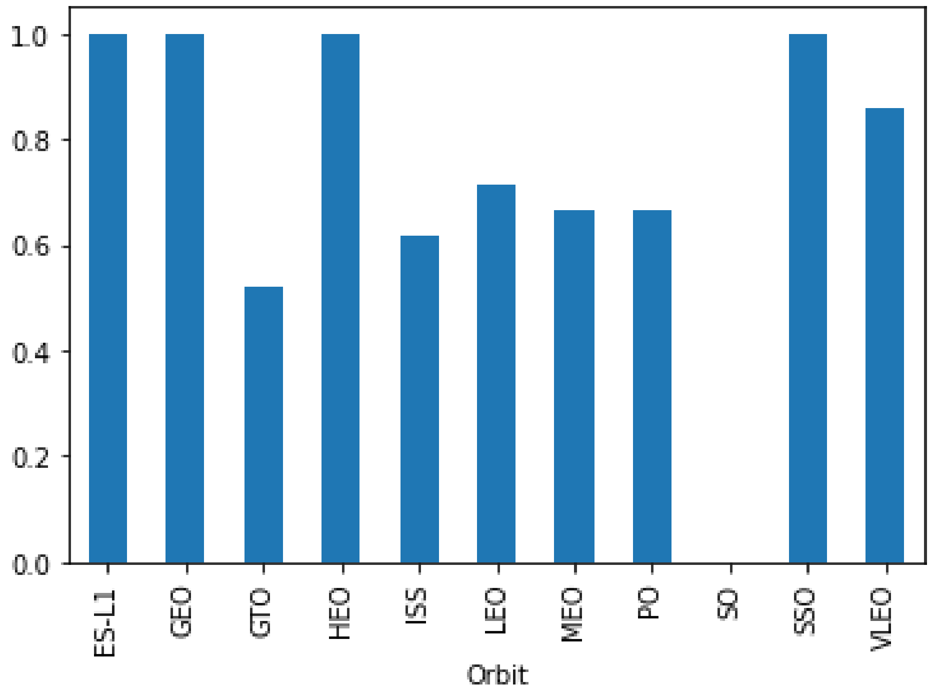
Flight Number vs. Launch Site

- The higher the flight number the higher success rate
- Low flight numbers mainly launch from CCAFS SLC 40



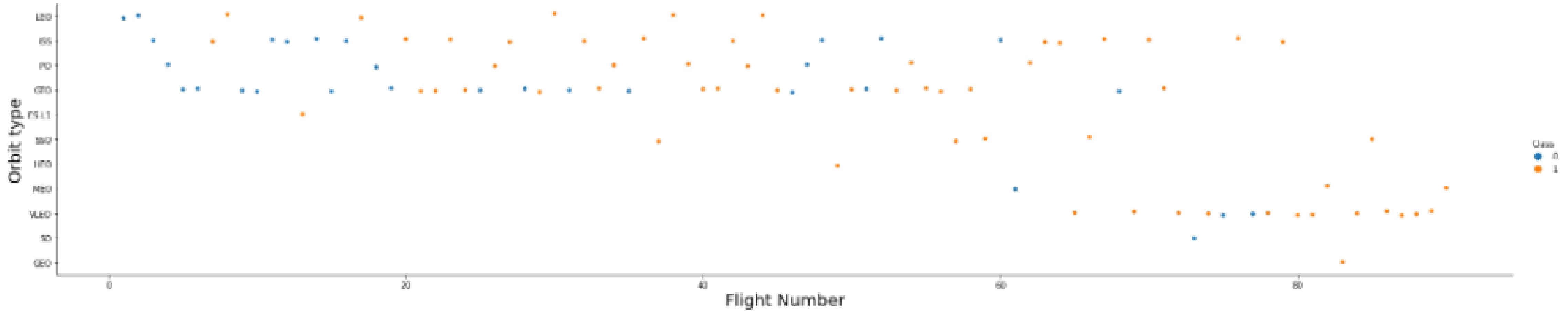
Payload vs. Launch Site

- With high payloads the success rate is higher
- Highest payloads launch from CCAFS SLC 40 mainly
- Low payloads below 2000 dont launch from KSC LC 39A but rather mainly from CCAFS SLC 40



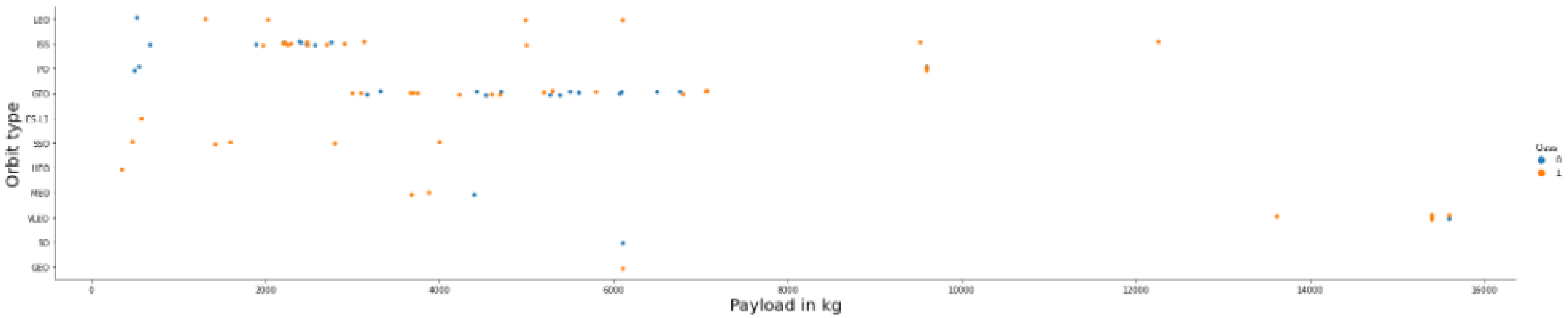
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, and SSO have the highest success rate of 1



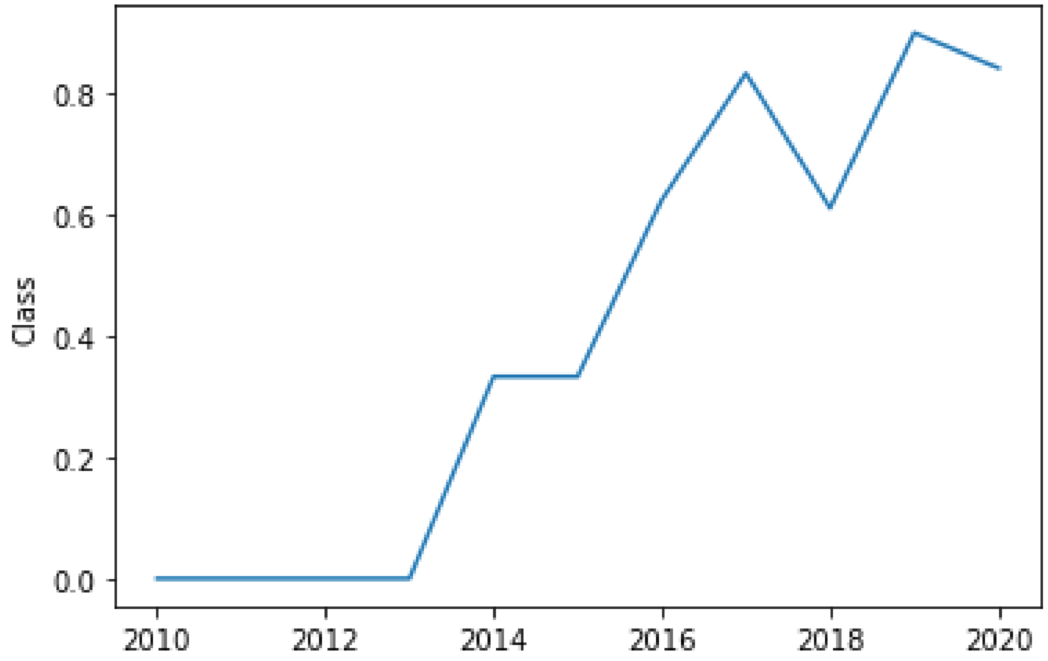
Flight Number vs. Orbit Type

- High flight numbers mainly go to GEO,SO,VLEO,MEO,HEO,SOO orbits
- VLEO orbit has a high success rate
- The higher the flight number the higher success rate for LEO orbit



Payload vs. Orbit Type

- High payload don’t go to SSO,MEO,SO,GEO,GTO orbits
- In VLEO orbit high payload results in failed mission
- ISS orbit has higher SR with heavier payloads



Launch Success Yearly Trend

- Success rate is higher as the time goes on

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

```
%%sql
select distinct(LAUNCH_SITE) from SPACEXTABLE
```

Launch Site Names Begin with 'CCA'

- Selects all distinct values from Launch_Site column in the table

```
%%sql
select * from SPACEXTABLE
where LAUNCH_SITE like 'CCA%' limit 5
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.bluemix.net:50000/BLUDB
Done.
```

DATE	time__utc_	booster_version	launch_site	payload_mass__kg	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	None	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	None	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	None	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	None	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	None	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Launch Site Names Begin with 'CCA'

- The limit 5 key ensures we only display 5 records
- LIKE CCA% ensures that the launch_site name begins with CCA

```
%%sql
select sum(PAYLOAD_MASS__KG_) as "NASA (CRS)_payload_mass" from SPACEXTABLE
where CUSTOMER = 'NASA (CRS)'
group by CUSTOMER
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.bluem
Done.
```

NASA (CRS)_payload_mass
45596

Total Payload Mass

- Sum() sums the values where customer is NASA(CRS) and is grouped by customer

```
%%sql
select avg(PAYLOAD_MASS__KG_) as "F9 v1.1 payload" from SPACEXTABLE
where BOOSTER_VERSION = 'F9 v1.1'
group by BOOSTER_VERSION
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-
Done.
```

F9 v1.1 payload
2928.400000

Average Payload Mass by F9 v1.1

- Selects the average (avg()) of payload where the vlaue in the booster version column is F9 v1.1

```
%%sql
select min(DATE) as "first successful landing outcome in ground pad" from SPACEXTABLE
where LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.bluemix.net:500
Done.
```

first successful landing outcome in ground pad
2015-12-22

First Successful Ground Landing Date

- Selects the minimum (min()) date (so the oldest date) from the table where the value in the landing_outcome column is "success"

```
%%sql

select BOOSTER_VERSION,PAYLOAD_MASS__KG_,LANDING__OUTCOME from SPACEXTABLE
where PAYLOAD_MASS__KG_ between 4000 and 6000
and LANDING__OUTCOME = 'Success (drone ship)'
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.bluei
Done.
```

booster_version	payload_mass__kg_	landing__outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Successful Drone Ship Landing with Payload between 4000 and 6000

- Selects all the variables (BV,PMKG,LO) from the table under the condition that payload mass value is between 4k and 6k and landing outcome column value is "Success (drone ship)"

```
%%sql
select MISSION_OUTCOME, count(*) as count from SPACEXTABLE
group by MISSION_OUTCOME
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.servi
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- Selects the count of mission outcomes per category in the mission-outcome column

```
%%sql
select BOOSTER_VERSION,PAYLOAD_MASS__KG_ from SPACEXTABLE
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.bluem
Done.
```

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Boosters Carried Maximum Payload

- Selects the names of BV and values of payload mass from the table. Then a subquery is used to selct the maximum(max()) payload mass value from the column

```
%%sql
select LANDING__OUTCOME,BOOSTER_VERSION,LAUNCH_SITE,DATE from SPACEXTABLE
where LANDING__OUTCOME = 'Failure (drone ship)' and year(DATE)= '2015'
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.services.eu-gb.blue
Done.
```

landing__outcome	booster_version	launch_site	DATE
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-01-10
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

2015 Launch Records

- Selects LO,BV,LS and date from the table where the landing outcome on a drone ship was a failure and the year of the date was 2015. Uses and clause to have both outcome and year as requirements for value to be put into the queried table

```
%%sql
select LANDING__OUTCOME,count(*) as count from SPACEXTABLE
where DATE between '2010-06-04' and '2017-03-20'
group by LANDING__OUTCOME
order by count desc
```

```
 * ibm_db_sa://nhb20692:***@dashdb-txn-sbox-yp-lon02-13.serv
Done.
```

landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

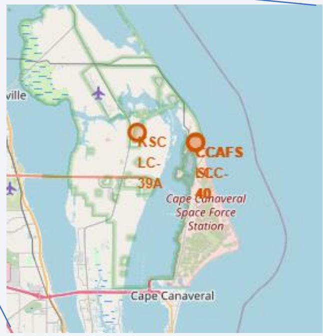
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- Selects the count of unique values per landing outcome in the given period (2010-06-04 and 2017-23-20), then is grouped by the landing outcome value and then ordered to descending so the first value shows the highest counts of landing outcome

Launch Sites Proximities Analysis

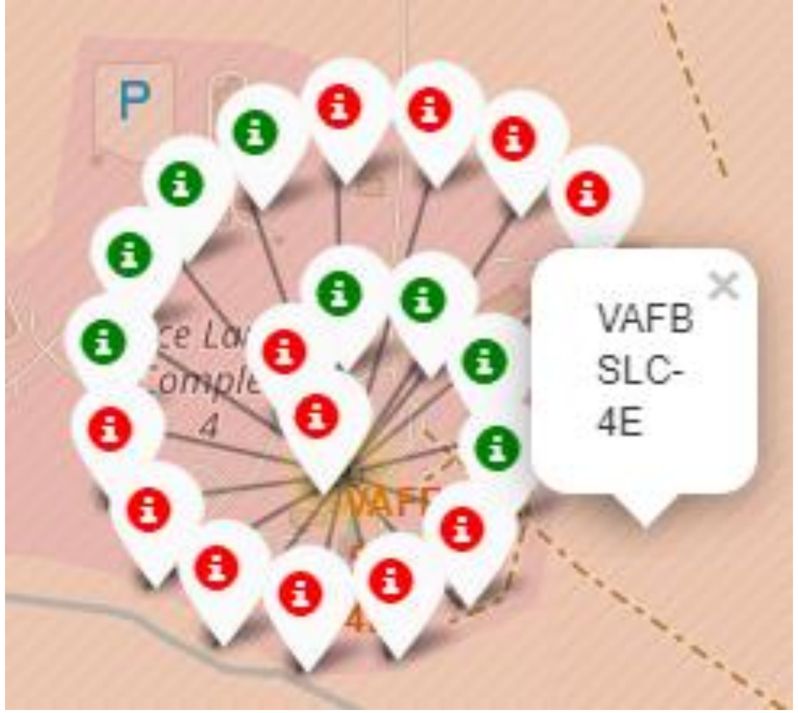
Folium Map SpaceX Launch Sites

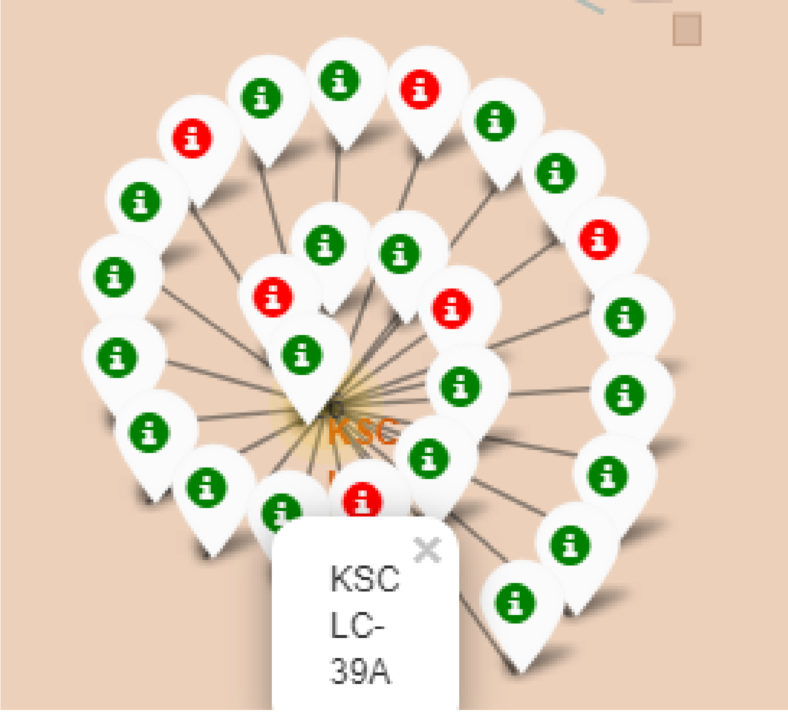
- All SpaceX launsites are located near oceans on both coasts of the USA.

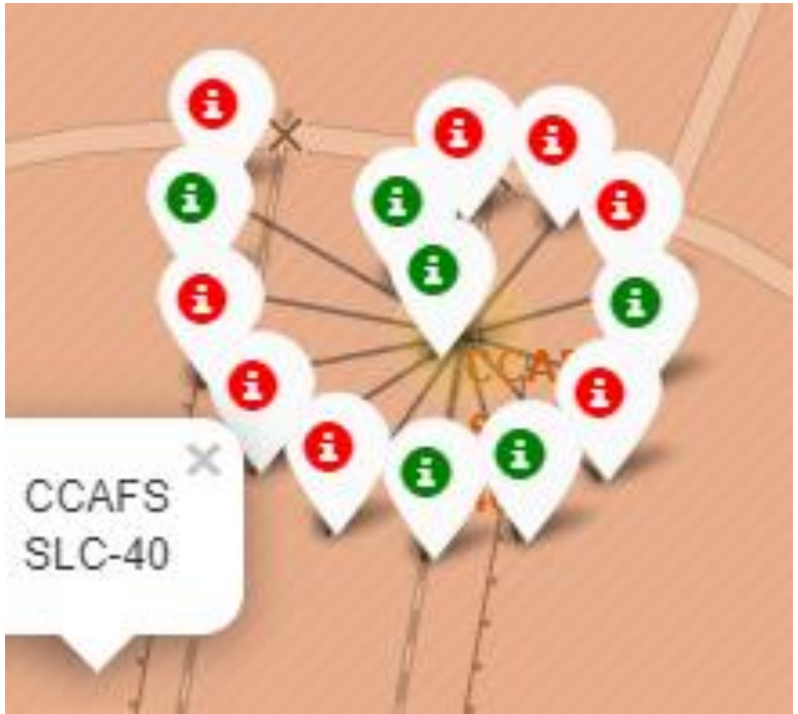


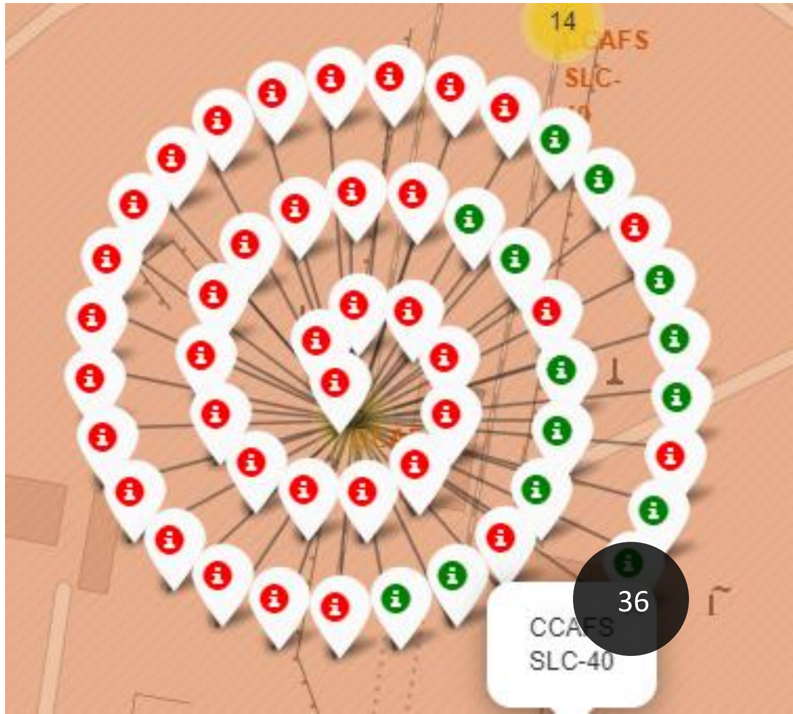
Launch Site Success Rate

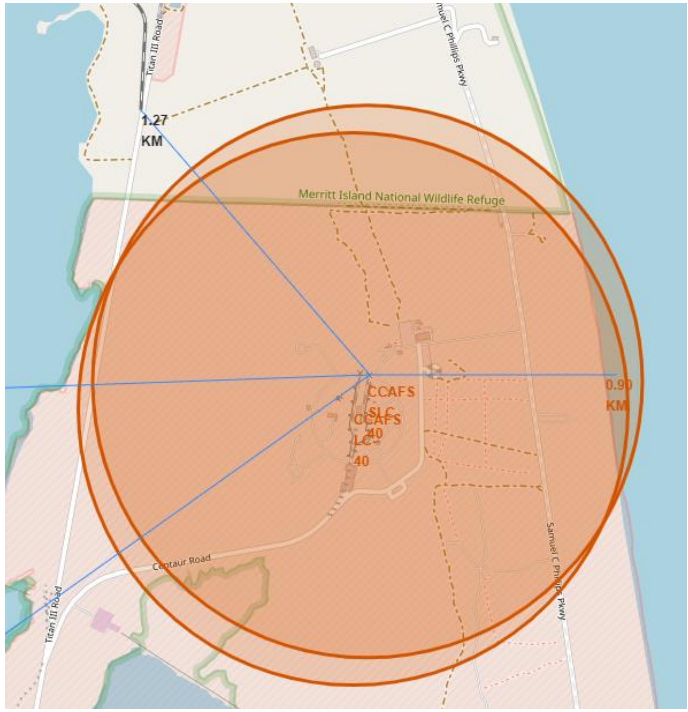
- Successful landing outcomes are labeled with a red marker and failed landings are labeled with a green marker
- From these screenshots we can deduct that the highest success rate is on the KSC LC-40 site

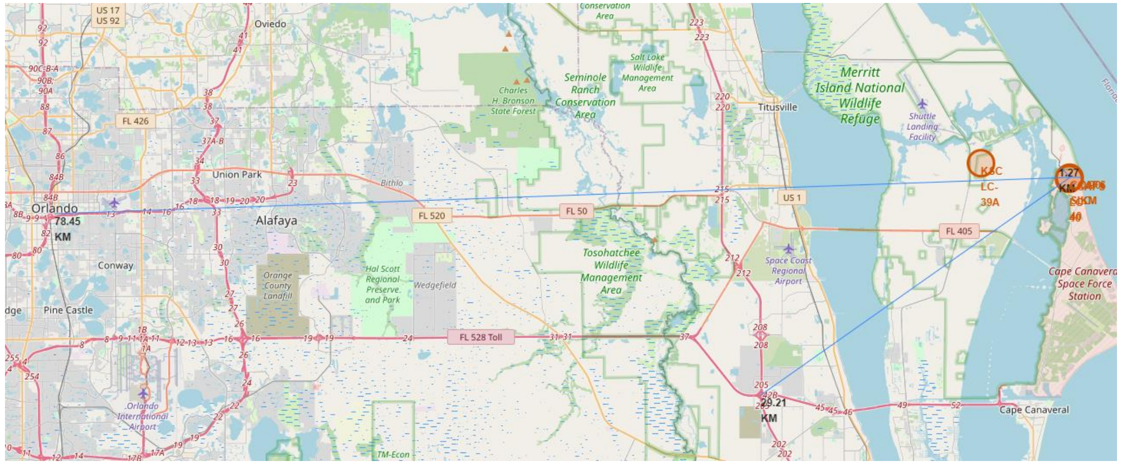












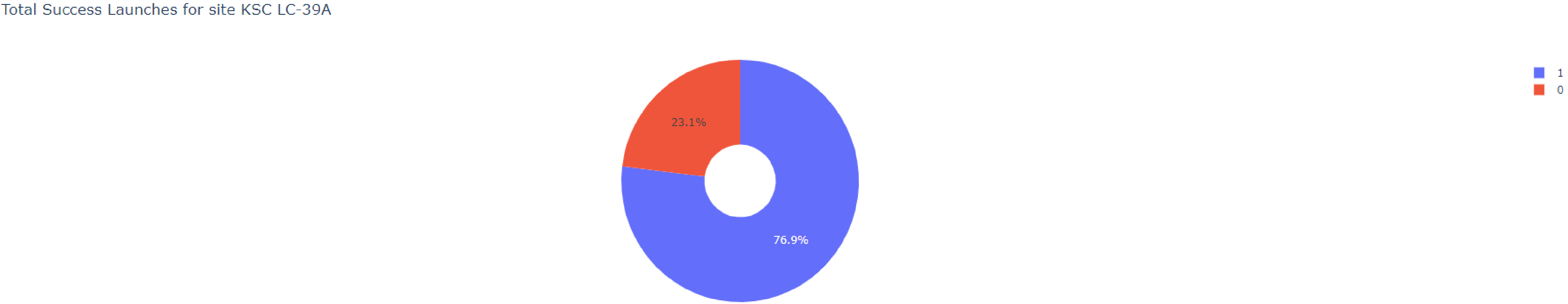
Launch Site Distances to Railway,Coast,City,Highway

- The distance from the CCAFS launch site to the closest railway is almost 1.3km
- The distance from the same site to nearest coast line is only 900m
- The distance to the highway is 29.22km
- The distance to Orlando city is 78.45km

Build a Dashboard with Plotly Dash

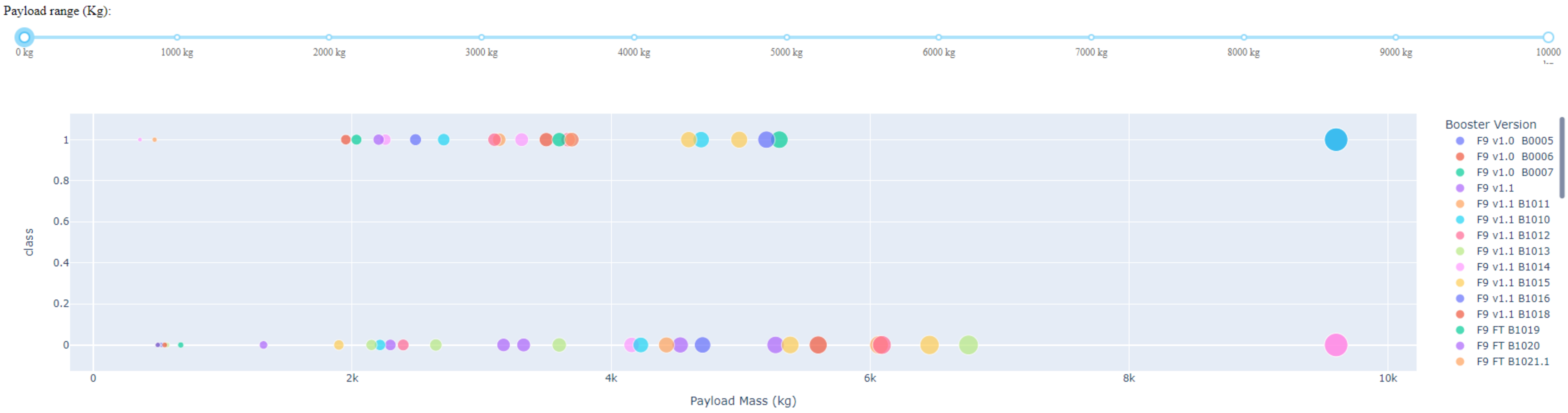
All sites success rate pie chart using Plotly Dash

- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot



Highest Success Rate Pie Chart

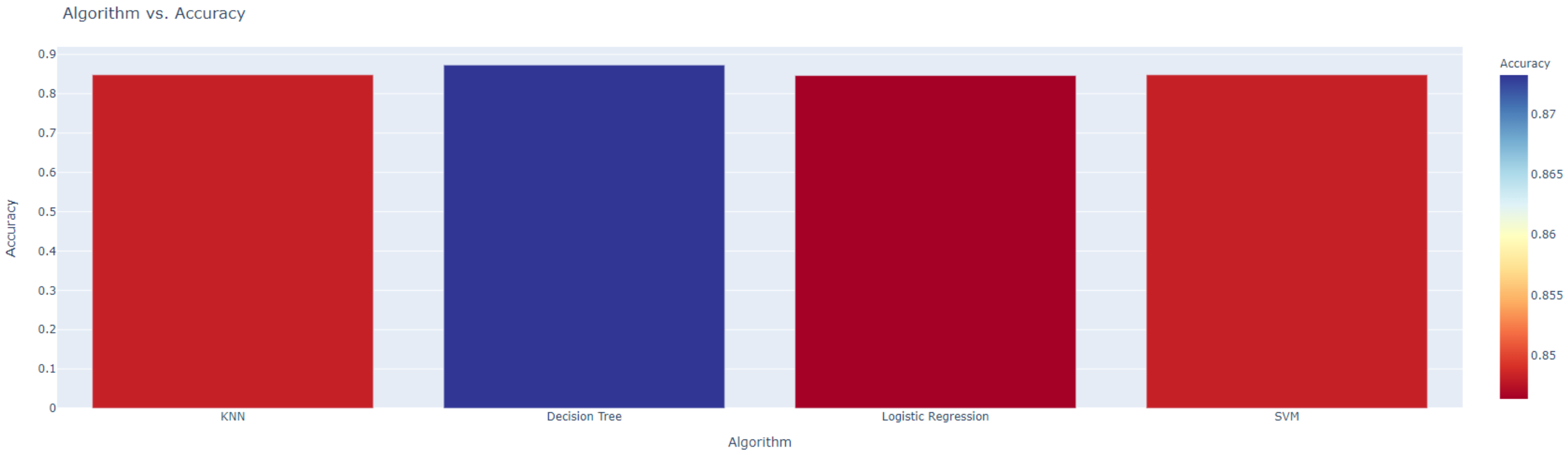
- Launch site named KSC LC-39A has the highest success rate of 76.9%



Payload vs. Launch outcome scatter plot

- The scatter plot shows the outcome to be either 1(success) or 0(fail) on the y-axis and also displays the different boosterversions with color gradient. The x-axis shows the payload, which can be adjusted on the slider above.
- Higher payloads have lowe succes rate

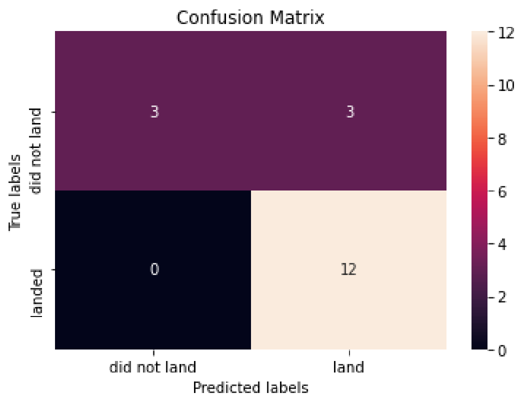
Predictive Analysis (Classification)



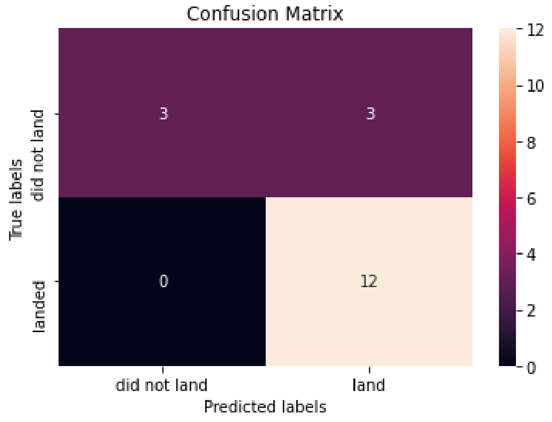
Classification Accuracy

- The Decision tree model had the best accuracy from all the models

LogReg

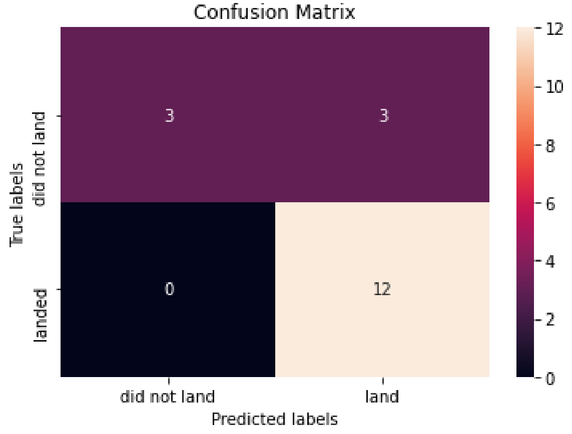


SVM



DecTree

KNN



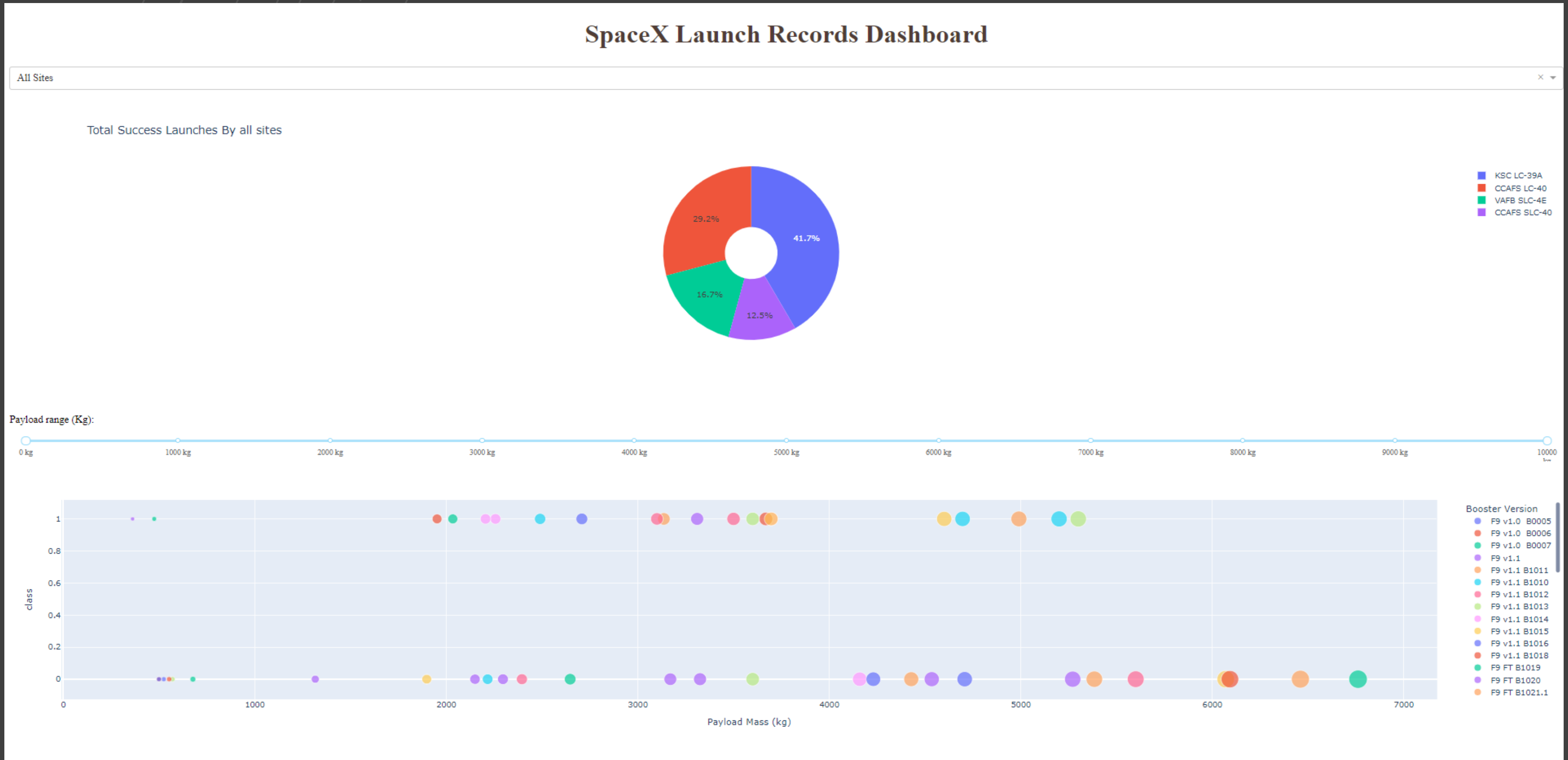
Confusion Matrices

- Even though we found that the decision tree model was best performing, all of the models have the same confusion matrix

Conclusions

- Orbits HEO,GEO,SSO,ES-L1 were found to have the highest success rates
- KSC LC-39A had the best launch succes rate from the three sites
- We found that as time goes on SpaceX launch success rate is getting better and better due to experience and trial and error finetuning
- The heavier the payload the higher the chance of failure
- Decision tree algorithm was found to be the best algorithm for machine learning

Appendix


SpaceX Launch Records Dashboard
All Sites
Total Success Launches By all sites
41.7%
29.2%
16.7%
12.5%
KSC LC-39A
CCAFS LC-40
VAFB SLC-4E
CCAFS SLC-40
Payload range (Kg):
Payload Mass (kg)
class
Booster Version
F9 v1.0 B0005
F9 v1.0 B0006
F9 v1.0 B0007
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1010
F9 v1.1 B1012
F9 v1.1 B1013
F9 v1.1 B1014
F9 v1.1 B1015
F9 v1.1 B1016
F9 v1.1 B1018
F9 FT B1019
F9 FT B1020
F9 FT B1021.1

Appendix

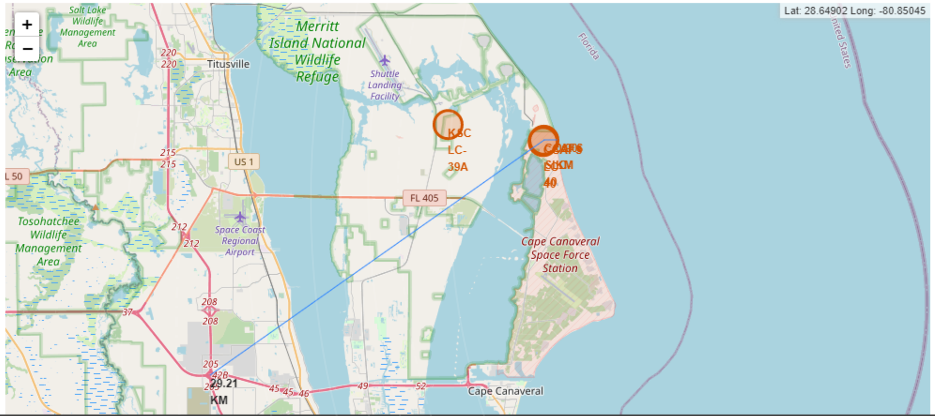
```
# find coordinate of railway point

distance_railway = calculate_distance(34.632834, -120.610746, 34.63632, -120.62383)


string = "{} Km".format(round(distance_railway,1))
print(string)
```

```
#highway
coordinates = [
    [28.56342, -80.57674],
    [28.411780, -80.820630]]

lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
distance = calculate_distance(coordinates[0][0], coordinates[0][1], coordinates[1][0], coordinates[1][1])
distance_circle = folium.Marker(
    [28.411780, -80.820630],
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#252526;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
        )
    )
site_map.add_child(distance_circle)
site_map
```




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

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
gs_cv = GridSearchCV(lr, parameters, scoring='accuracy', cv=10)
logreg_cv = gs_cv.fit(X_train, Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data a
validation data using the data attribute `best_score\_`.

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

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

TASK 5

Calculate the accuracy on the test data using the method `score`:

```
print("accuracy is: ", logreg_cv.score(X_test, Y_test))
```

```
accuracy is:  0.8333333333333334
```

Lets look at the confusion matrix:

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

```
algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM':svm
_cv.best_score_}
best_algorithm = max(algorithms, key= lambda x: algorithms[x])

print('The method which performs best is \"',best_algorithm,'\" with a score of',algorithms[best_algorithm])
```

```
The method which performs best is " Decision Tree " with a score of 0.8732142857142856
```

```
algdf = pd.DataFrame.from_dict(algorithms, orient='index', columns=['Accuracy'])
```

```
algdf = algdf.reset_index()
algdf.rename(columns = {'index': 'Algorithm'}, inplace = True)
```

```
import plotly.express as px
import plotly.graph_objects as go
fig = px.bar(algdf, x='Algorithm', y='Accuracy', hover_data=['Algorithm', 'Accuracy'], color='Accuracy', color_continuous_scale
='rdylbu')
fig.update_layout(title='Algorithm vs. Accuracy', xaxis_title='Algorithm', yaxis_title='Accuracy' )
fig.show()
```


Thank you