IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<AMAL MEJDOUB EP LOUKIL>
<12/15/2021>

Outline

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

Executive Summary

- In data science, data is the important component of the project. Once collected, a data scientist needs to process the data and get rid of irrelevant data. Some of the methodology I used in my project are data sampling and dealing with nulls. To understand and have a clear vision of the data, a data scientist needs to visualize data using plots, maps and dashboard. Then, a model also called machine learning needs to be build to predict results for future scenarios. Machine learning deals with numerical variables for best performance. One hot encoding is one of the method to convert categorical data into numerical data which I also used in my project to convert categorical columns values.
- In this project, I studied the SpaceX dataset and I confirm the following results: Most of the launch sites are close to cost. KSC LC-39 has the highest successful launches. There is a correlation between payload mass and success rate. If the payload mass is important >7k, then it is hard to reuse the first stage. All the models used in this project(Logistic regression, SVM, Decision Tree, K-nearest Neighbors) have the same test data accuracy.

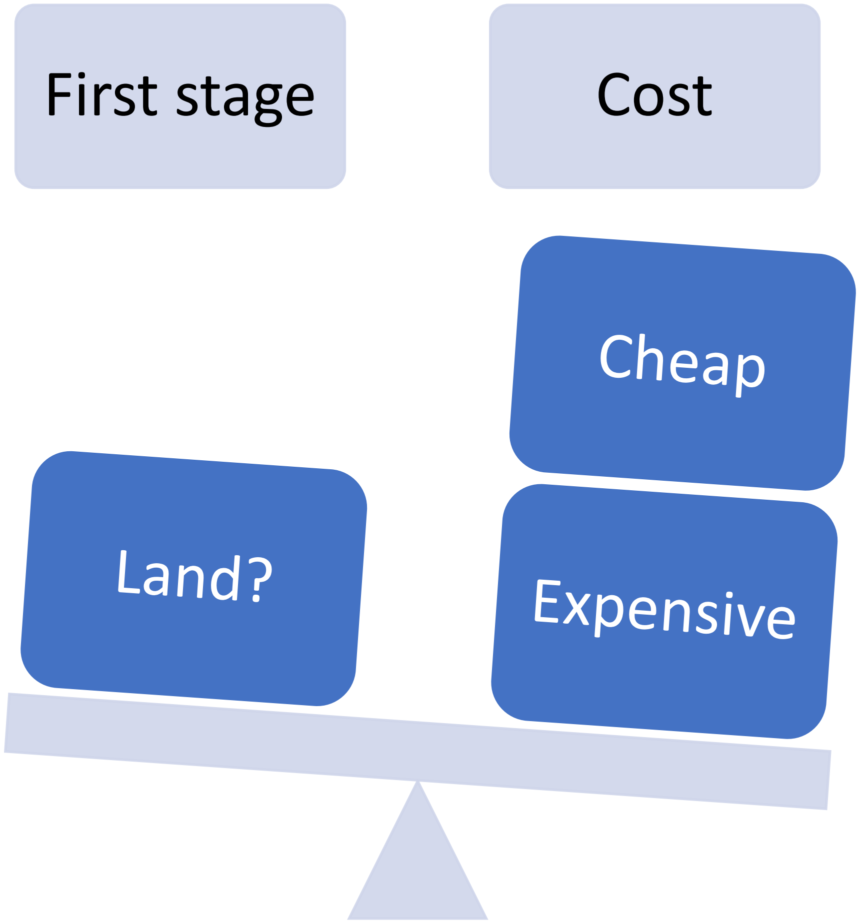
Introduction

Watching a rocket launch either live or on TV has always been a jaw-dropping. A new market called space tourism is growing. It refers to the activity of travelling into space for recreational purposes. Thus, many companies like virgin galactic, Bleu Origin are racing to offer affordable commercial human spaceflight. Perhaps, the most successful is SpaceX. Besides being the first private company to launch four people to orbit for three days, they succeed to send space craft to the international space station and to send satellites.
But did you wonder one day how much it costs to launch a rocket into space? These companies are competing to lower the cost of their launching: 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. How can SpaceX launches be so cheap? Much of the savings is because SpaceX can reuse the first stage for future launches. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

Problem Formulation

Data: Gathering information about SpaceX.

Project Goal: Determine if spaceX will reuse the first stage?



Flowchart1: Determine the cost depending on first stage

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  1- SpaceX Rest API

  2- Web scraping related wiki pages

- Perform data wrangling

  1- Wrangling data using API

  2- Sampling Data: Filter/Sample the data to remove Falcon1 launches

  3- Dealing with Nulls: Replace null values with mean value

Methodology

Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Flowchart2: Predictive analysis using classification models

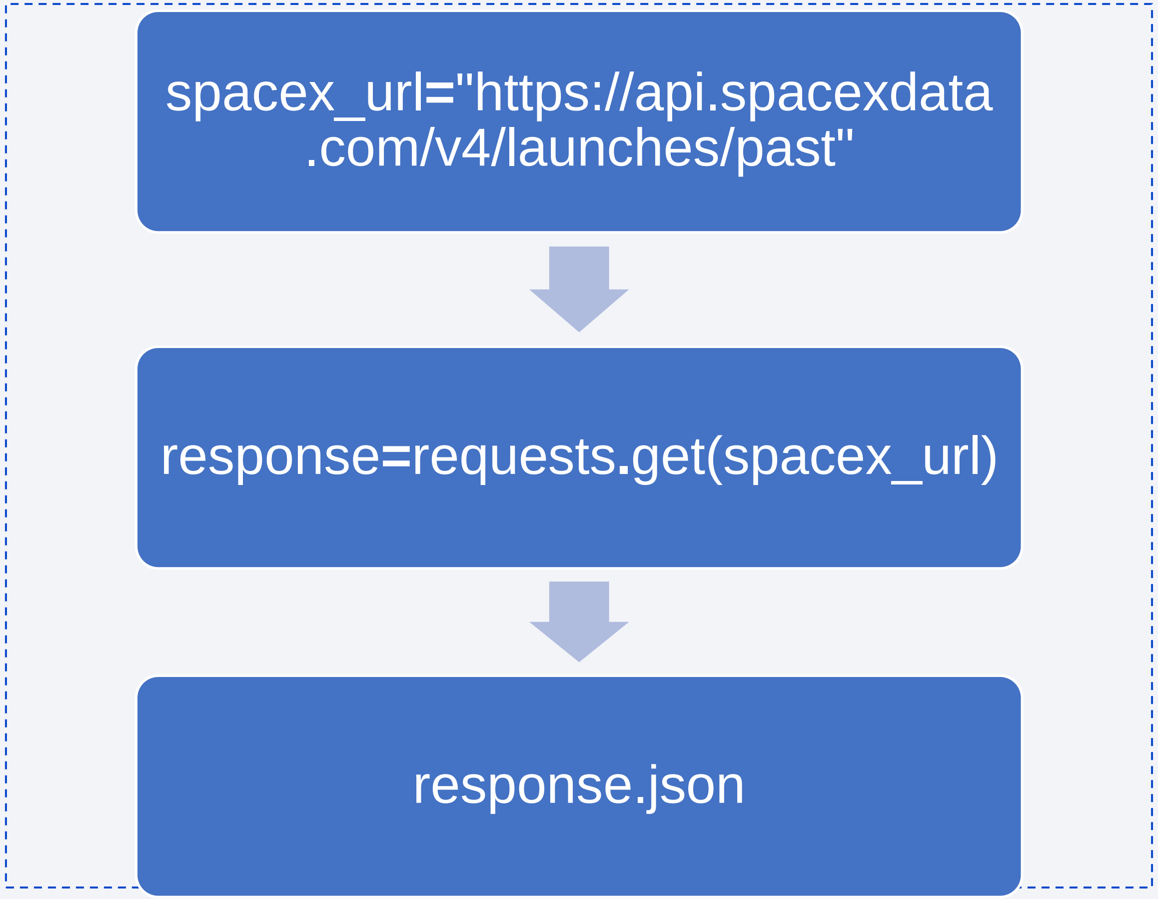
Data Collection

- The data used in this project is gathered from SpaceX Rest API and specifically from api.SpaceXdata.com/V4/launches/past endpoint to view SpaceX past launches. Another data source for obtaining Falcon 9 launches is web scraping related wiki pages. Next, I will show how i collected data from the two sources mentioned above using flowcharts.

Data Collection – SpaceX API

- Step 1: Use spacex_url to get past launches.
- Step 2: Perform a get request to obtain the launch data.
- Step 3: Convert response in the form of JSON

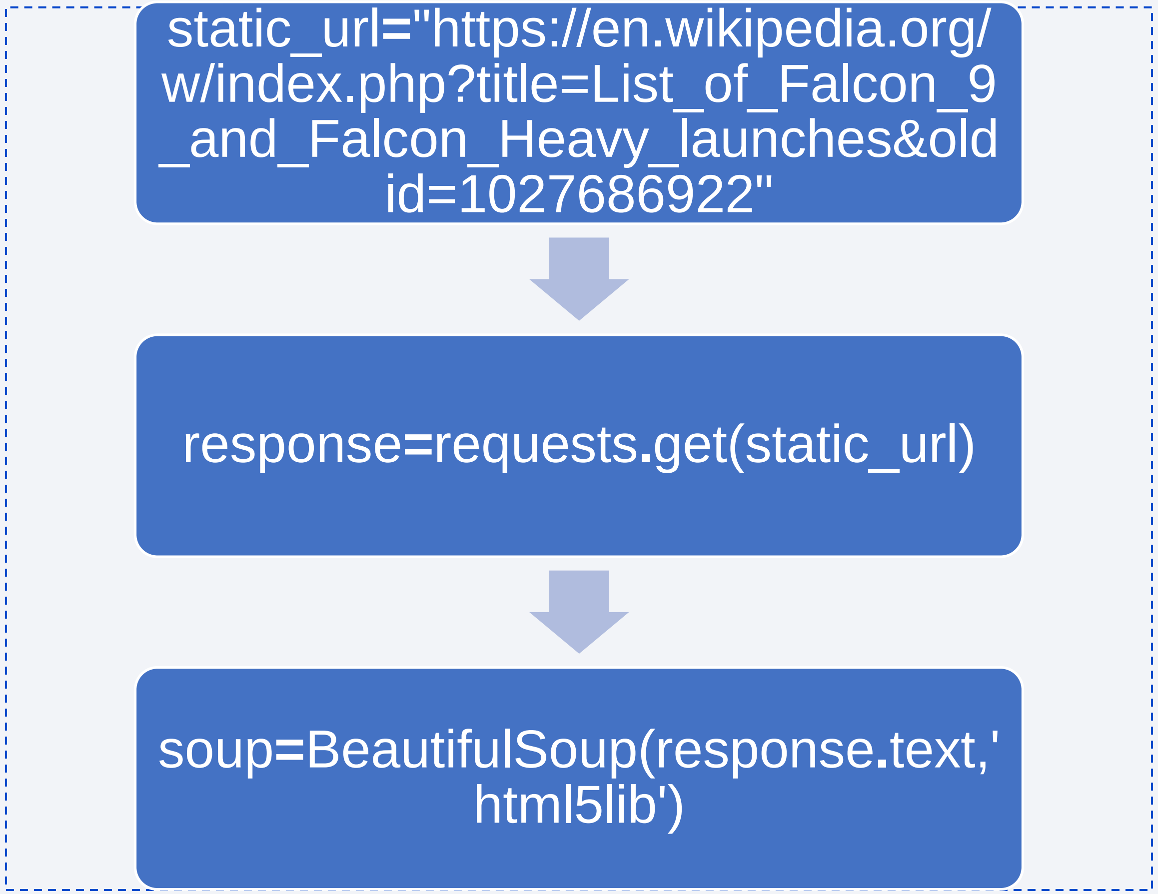
- https://github.com/mejdoub87/project-capstone/blob/master/the%20Data%20Collection%20API%20Lab.ipynb



Flowchart3: Steps to collect data from SpaceX API

Data Collection - Scraping

- Step 1: Use the List of Falcon 9 and Falcon Heavy launches wiki page.
- Step 2: Perform a get request to get the Falcon9 Launch HTML page, as an HTTP response.
- Step 3: Use BeautifulSoup to parse the HTML page to extract the data easily.
- https://github.com/mejdoub87/project-capstone/blob/master/data%20collection%20web%20scraping.ipynb



Flowchart4: Steps to collect data from wiki page

Data Wrangling using API

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

Out[12]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f50
1	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f50

Flowchart5: Data wrangling using API

Comments:

I used the json_normalize function to convert the JSON to a dataframe. More specifically, it will allow to normalize the structured json data into a flat table as shown in the flowchart.

Data Wrangling: Sampling Data

```
data_falcon9=launch_data[launch_data['BoosterVersion']!='Falcon 1']
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	l
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-1
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1060	-
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1058	-
91	88	2020-	Falcon 9	15600.0	VLEO	KSC LC 39A	True	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	9	B1051	-

Flowchart6: Sampling Data

Comments:

In this project, I am interested in Falcon 9 launches data. To do so, I used the code line shown in the flowchart to filter the dataframe based on the condition launch_data['BoosterVersion']!='Falcon 1’

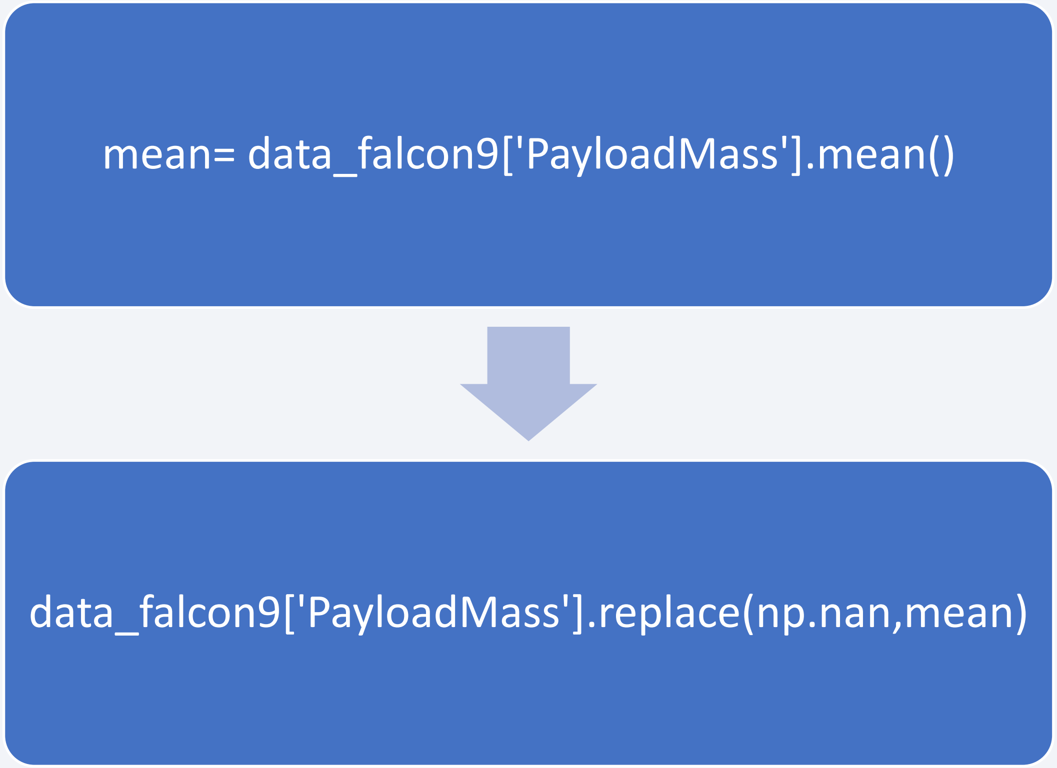
This will only return launches data corresponding to Falcon 9.

Calculate below the mean for the PayloadMass

Data Wrangling: Dealing with nulls

- Step 1: I calculate the mean of the column payloadMass.
- Step 2: I replaced np.nan values with its means values.

- **Github URL:** https://github.com/mejdoub87/project-capstone/blob/master/Data%20wrangling.ipynb



Flowchart7: Dealing with nulls

EDA with Data Visualization

- The charts plotted in this project are :

  1- Scatter plot: Allow to find correlation between two variables.

  2- Bar chart: Allow to compare the success rate of each orbit.

  3- Line chart: Show the success rate over the years.

- GitHub URL:https://github.com/mejdoub87/project-capstone/blob/master/EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

The SQL queries I performed in this project are:

- Find the names of the unique launch sites
- Find 5 records where launch sites begin with `CCA`
- Calculate the total payload carried by boosters from NASA
- Calculate the average payload mass carried by booster version F9 v1.1
- Find the dates of the first successful landing outcome on ground pad
- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

EDA with SQL

- Calculate the total number of successful and failure mission outcomes
- List the names of the booster which have carried the maximum payload mass
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the 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
- GitHub URL: https://github.com/mejdoub87/project-capstone/blob/master/EDA%20with%20SQL%20lab.ipynb

Build an Interactive Map with Folium

- The map objects I created in this project are markers, circles, clusters and lines. Then I added each object to the folium map.
- **Why I used these objects?**

1- Line object: To trace the distance between a launch site and its proximities.

2- Cluster object: To group the markers into different clusters. Each cluster represents the number of success/failed launches for each site.

3- Marker/circle objects: To mark each launch site in the folium map and give it a color.

- **GitHub URL :** https://github.com/mejdoub87/project-capstone/blob/master/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- **In order to successfully accomplished the Dashboard, I added the following tasks:**

1- Dropdown menu using dcc.dropdown. Then , I populated with the values: All, CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40 and I set the default value to 'ALL'.

2- Range slider using dcc.RangeSlider and set the default values to min_payload and max_payload.

3- Scatter plot to study the correlation between the playload mass and the success rate for all sites.

4- Pie chart to represent the success rate for all sites.

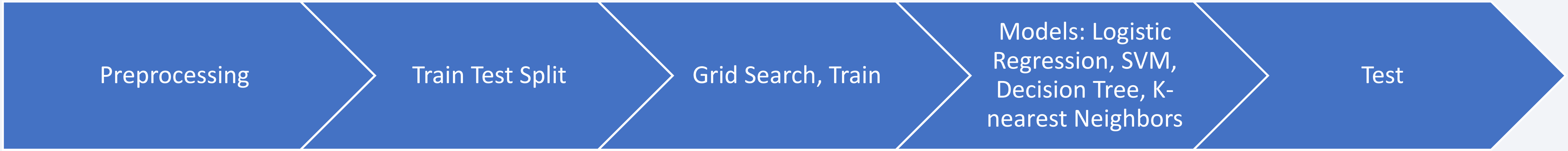
5- @app.callback function for the dropdown menu and the slide ranger.

6- get_scatter_chart and get_pie_chart functions to return the pie chart/scatter chart depending on the values choosed on the dropdown menu and the range slider.

- **GitHub URL:** https://github.com/mejdoub87/project-capstone/blob/master/visualization.ipynb

Predictive Analysis (Classification)

- Model development:



- I choose to show the **Logistic Regression model**, and I followed the same steps to do for the other models as shown in the **screenshot**.

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

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params\_` and the accuracy on the 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
```

Results

- **Test Data Accuracy for each model screenshots:**

Logistic Regression

```
print('test data accuracy:',logreg_cv.score(X_test, Y_test))
test data accuracy: 0.8333333333333334
```

SVM

```
print("test data accuracy:", svm_cv.score(X_test, Y_test))
test data accuracy: 0.8333333333333334
```

```
print("test data accurracy:", tree_cv.score(X_test, Y_test))
test data accurracy: 0.8333333333333334
```

Decision tree

```
print("test data accurracy:",knn_cv.score(X_test, Y_test))
test data accurracy: 0.8333333333333334
```

K-nearest neighbors

- All the models Logistic Regression, SVM, Decision Tree, K-nearest neighbors perform the same. In other word, they give the same test accuracy result which is equal to 0.83.
- GitHub URL: https://github.com/mejdoub87/project-capstone/blob/master/machine%20lab.ipynb

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Code Snippet:

```
# 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.scatterplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

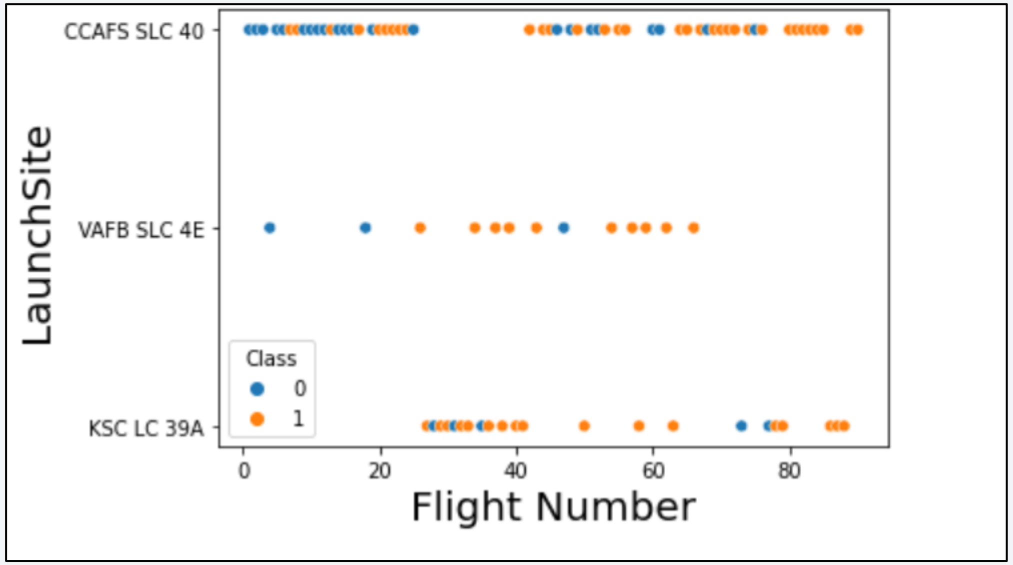


Fig1: Scatter point Flight Number/Launch Site

Result: The launch sites KSC LC-39A and CCAFS LC-40 has more successful rate when the flight number is grater then 80.

Payload vs. Launch Site

Code Snippet:

```
# 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 valu
e
sns.scatterplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```

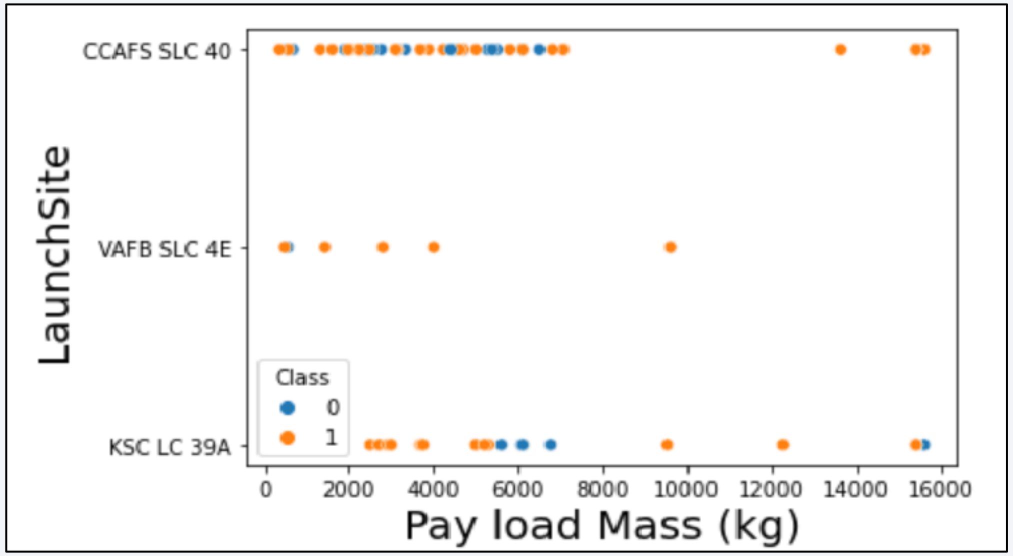


Fig2:Scatter plot of Payload vs. Launch Site

Result: For the VAFB-SLC 4E launch site, there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type

Code Snippet:

```
# HINT use groupby method on Orbit column and get the mean of Class column
group= df.groupby('Orbit').agg(success_rate=("Class",'mean'))
group = group.reset_index()
sns.barplot(x="Orbit", y="success_rate", data=group)
```

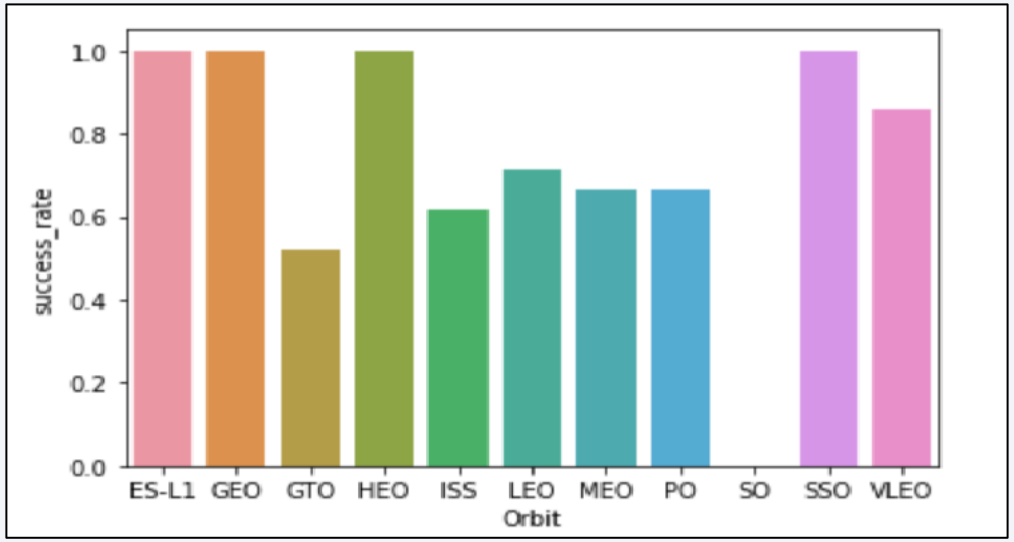


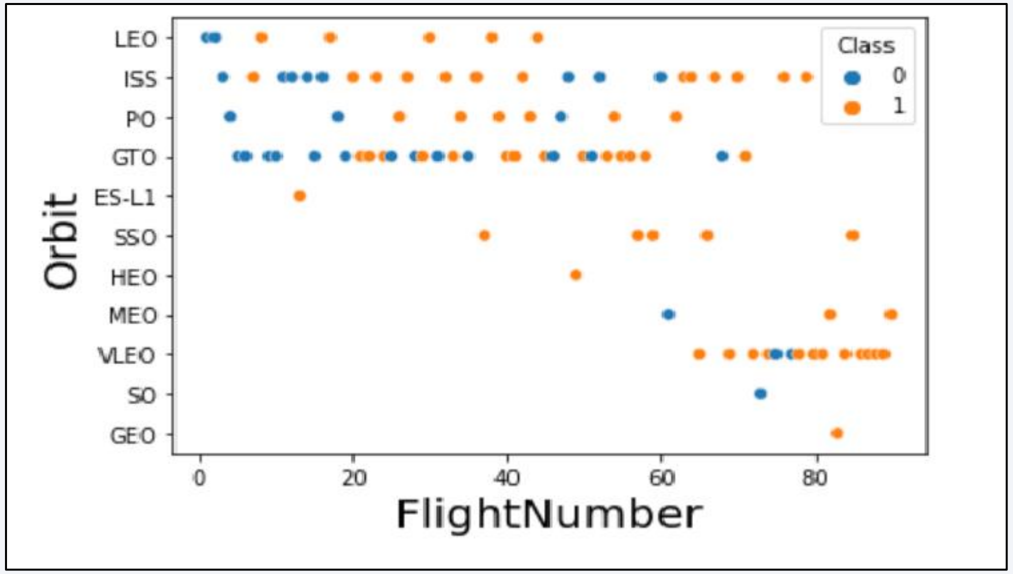
Fig3:Bar chart for the success rate of each orbit type

Result: ES-L1, GEO, HEO and SSO orbits have high success rate.

Flight Number vs. Orbit Type

Code Snippet:

```
# 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.scatterplot(y="Orbit", x="FlightNumber", hue="Class", data=df)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Result: In the ISS orbit, the success appears when the flight number increases (flight number >60).

Fig4: Scatter point of Flight number vs. Orbit type

Payload vs. Orbit Type

Code Snippet:

```
# 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.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

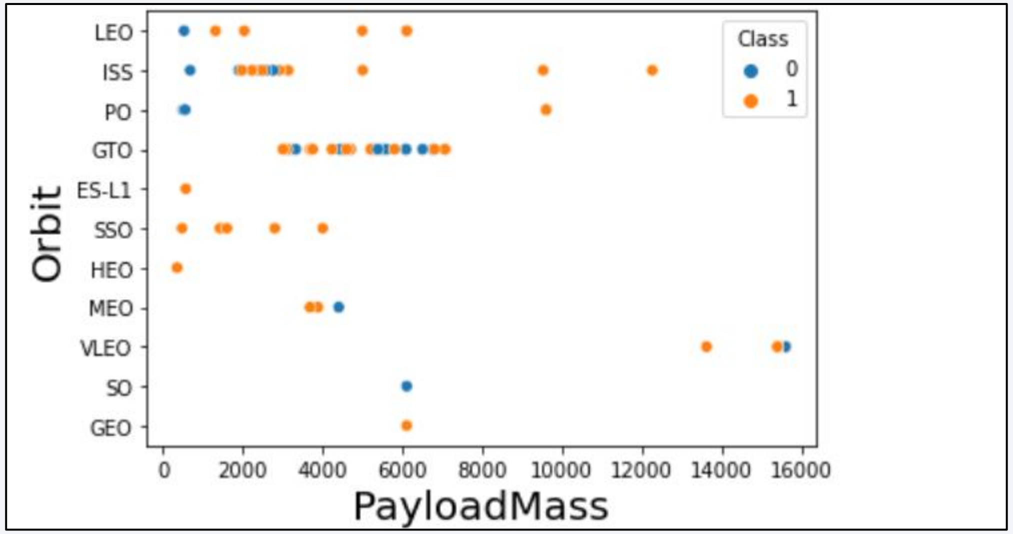


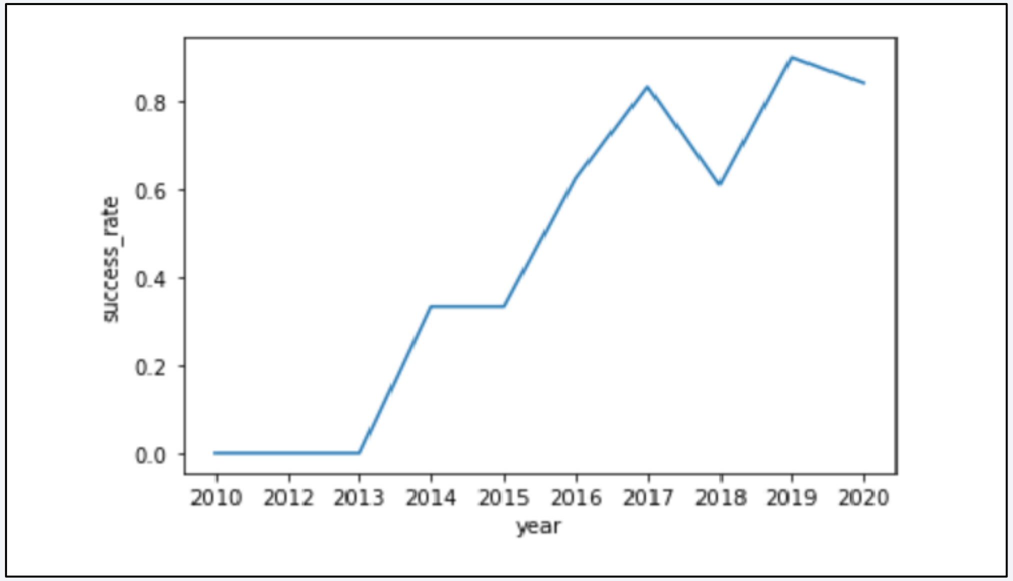
Fig5: Scatter point of payload vs. orbit type

Result: With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

Launch Success Yearly Trend

Code Snippet:

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
list=Extract_year(df['Date'])
df1 = pd.DataFrame(list, columns =['year'])
df1['Class']=df['Class']
group=df1.groupby('year').agg(success_rate=("Class",'mean'))
group = group.reset_index()
sns.lineplot(data= group ,x= 'year', y='success_rate')
```



Result: you can observe that the success rate since 2013 kept increasing till 2020.

Fig6: Line chart of yearly average success rate

All Launch Site Names

Code Snippet: Find the names of the unique launch sites

```
In [30]: %sql select distinct(Launch_Site) from SPACEXDATASET

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
Out[30]:
```

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

Comments: The Keyword "distinct" in the SQL query will filter the results and will allow to eliminate the duplicate. In our case, the query returns the four launch sites: CCAFS LC-40, CCAFS SLC-40, KSC LC-39A and VAFB SLC-4E.

Launch Site Names Begin with 'CCA'

Code Snippet: Find 5 records where launch sites begin with `CCA`

```
n [31]:  %sql select Launch_Site from SPACEXDATASET where Launch_Site like 'CCA%' limit 5

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
ut[31]:
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Comments: I filtred my results using “limit” keyword to only show the first five rows that correspond to the launch site beginning with ‘CCA’.

Total Payload Mass

Code Snippet: Calculate the total payload carried by boosters from NASA

```
[32]: %sql select sum(PAYLOAD_MASS__KG_) as Total_playload from SPACEXDATASET where Customer like 'NASA (CRS)'

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

[32]:

total_playload
45596

Comments: I used the aggregate function "sum" to calculate the total payload carried by boosters from NASA.

Average Payload Mass by F9 v1.1

Code Snippet: Calculate the average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as Average from SPACEXDATASET where Booster_Version like 'F9 v1.1'
```

```
 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

average
2928

Comments: I used the aggregation function "average" to calculate the average payload mass carried by booster version F9 v1.1.

First Successful Ground Landing Date

Code Snippet: Find the dates of the first successful landing outcome on ground pad

```
%sql select min(DATE) as min_date from SPACEXDATASET where LANDING__OUTCOME like 'Success (ground pad)'
```

```
 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

min_date
2015-12-22

Comments: I used 'min' function to find the dates of the first landing outcome on ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

Code Snippet: List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql select BOOSTER_VERSION from SPACEXDATASET where LANDING__OUTCOME like 'Success (drone ship)' and (PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000)

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Comments: I used "between value 1 and value 2" keyword to filter my results and only return the names of boosters where landing_outcome like "success (drone ship) and value 1 equals to 4000 and value 2 equals to 6000.

Total Number of Successful and Failure Mission Outcomes

Code Snippet: Calculate the total number of successful and failure mission outcomes

```
%sql select Mission_Outcome, count(Mission_Outcome) as count from SPACEXDATASET Group by Mission_Outcome
```

```
 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Comments: I used 'Group by' on the column mission_outcome to group the identical data into groups. In this query, I have 3 groups Failure(in flight), success and success(payload status unclear), then I applied the count function to calculate the total number of successful and failure mission outcomes.

Boosters Carried Maximum Payload

Code Snippet: List the names of the booster which have carried the maximum payload mass

```
%sql  select Booster_Version from SPACEXDATASET where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXDATASET)
```

```
 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Comments: I used the following subquery(select max(payload_mass_kg) from spacexdataset) to return the maximum payload mass then I passed this value to the where clause of the main query to return the names of the booster with maximum payload mass.

2015 Launch Records

Code Snippet: List the failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015:

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

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

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

Comments: I used "year(date)" to extract the year from the date. In my query, I assigned year(date) to 2015 and passed to the where clause to get the list od failed landing_outcomes in drone ship.

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

Code Snippet: 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

```
%sql SELECT LANDING__OUTCOME, count(LANDING__OUTCOME) as count from SPACEXDATASET where DATE between '2010-06-04' and '2017-03-20' and LANDING__OUTCO

 * ibm_db_sa://nxy06270:***@2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:30756/bludb
Done.
```

landing__outcome	COUNT
Success (ground pad)	3
Failure (drone ship)	5

Comments: I first used the Group by clause to group the data into landing_outcome groups. Then, I used the order by desc to sort the results in the ascending order.

Section 4

Launch Sites Proximities Analysis

Mark all launch sites on a map

Finding: Most of the sites are in very close proximity to coast.

Mark the success/failed launches for each site on the map

Comments: I used clusters to group the markers into different clusters. Each cluster represents the number of success/failed launches for each site. The site VAFB SLC-4E has a total of 10 launches while KSC LC-39A has a total of 46 launches.

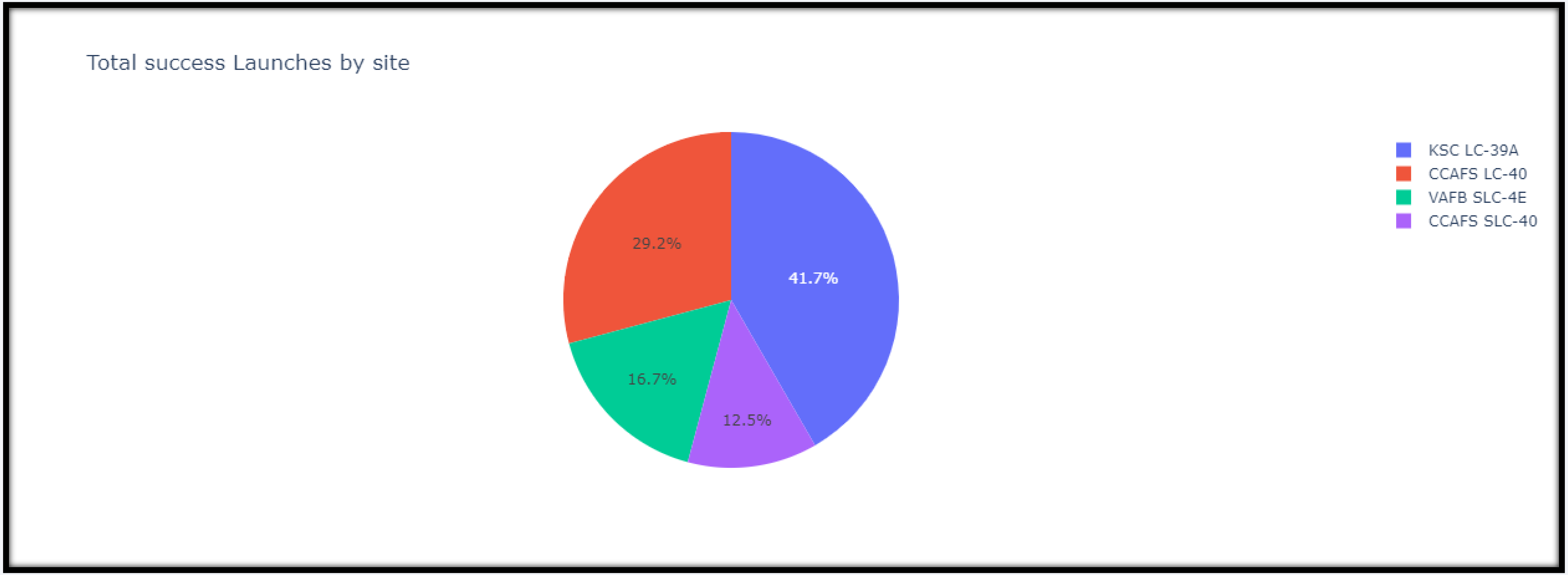
Calculate the distance between a launch site and coastline

Finding: I calculate the distance between the site VAFB SLC-4E and the coastline coordinate [34.58496, -75.64484]. The distance is equal to 4081. I also found that the site KSC LC-39A is almost as the same distance to railways and highways.

Section 5

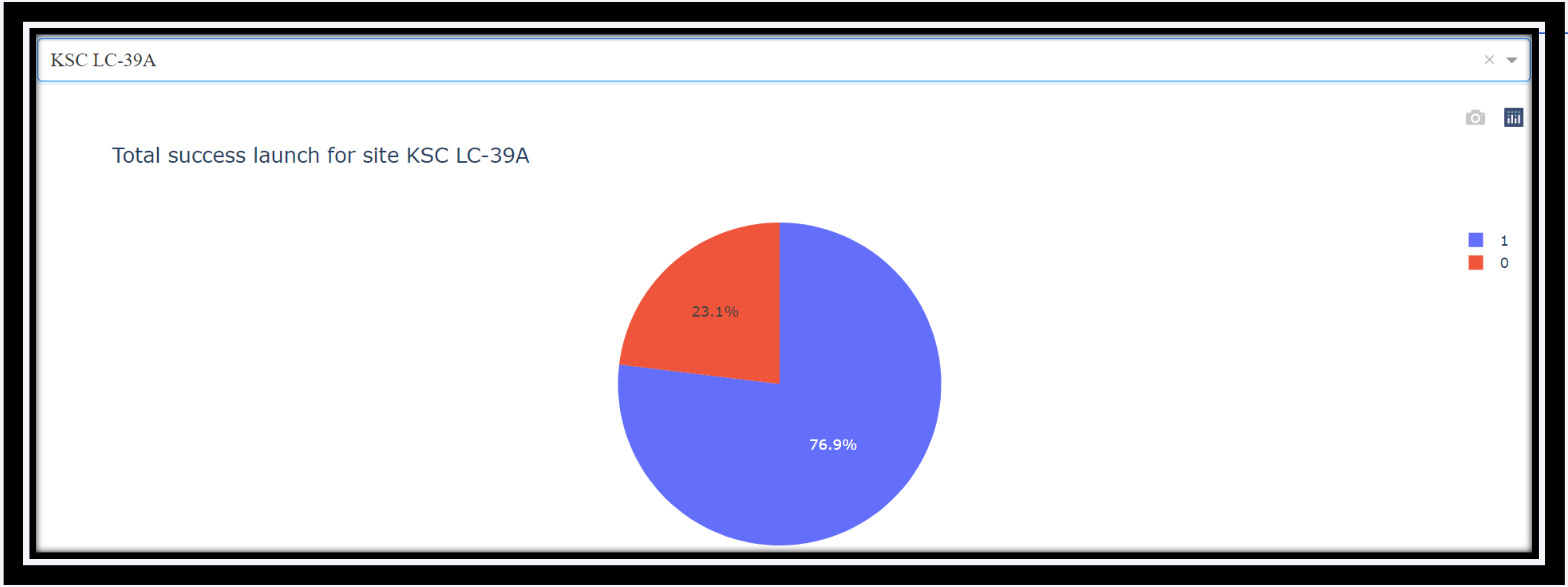
Build a Dashboard with Plotly Dash

Launch success count for all sites



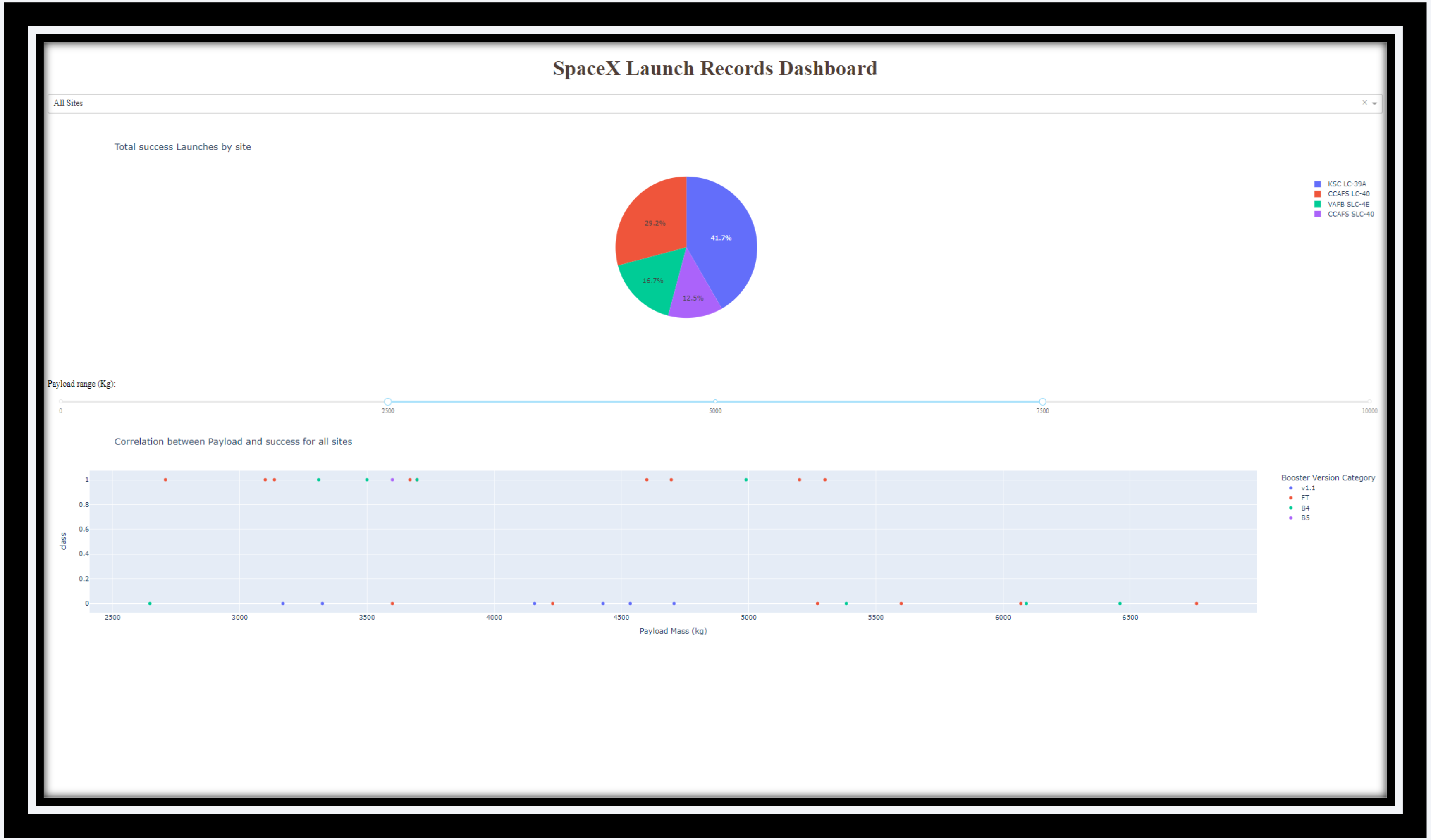
Finding: As we can see from the pie chart, KSC LC-39A site has the more successful launches.

Launch site with highest launch success ratio



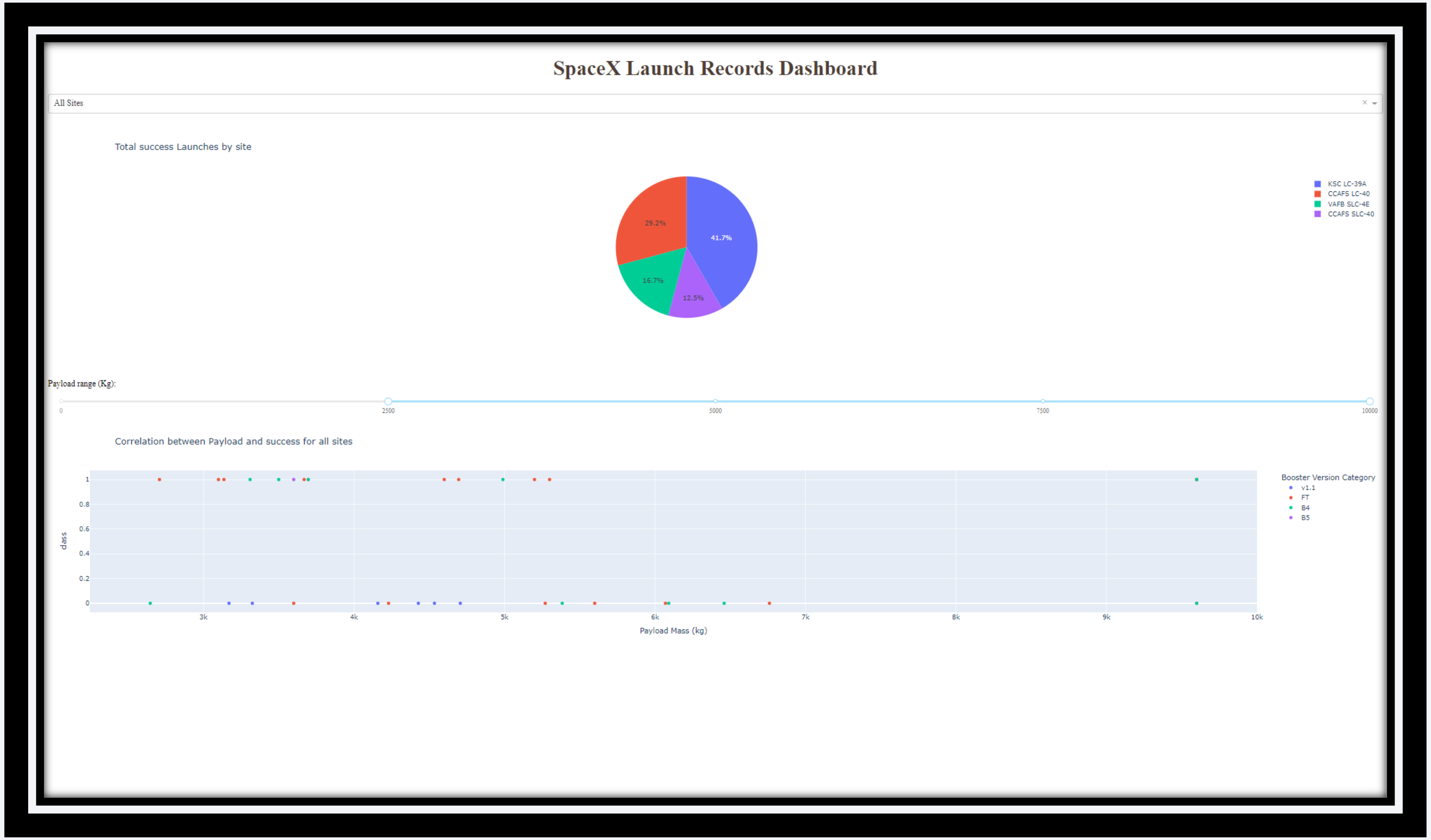
Finding: KSC LC-39A has the highest launch success ratio.

Correlation between payload and success rate for all sites range slider [2500,7500]



Finding: For the payload range slider between 2500 and 7500, the boosters' versions v1.1, FT, B4,B5 for all sites will not land successfully when the payload mass is high.

Correlation between payload and success rate for all sites range slider [2500, 10000]

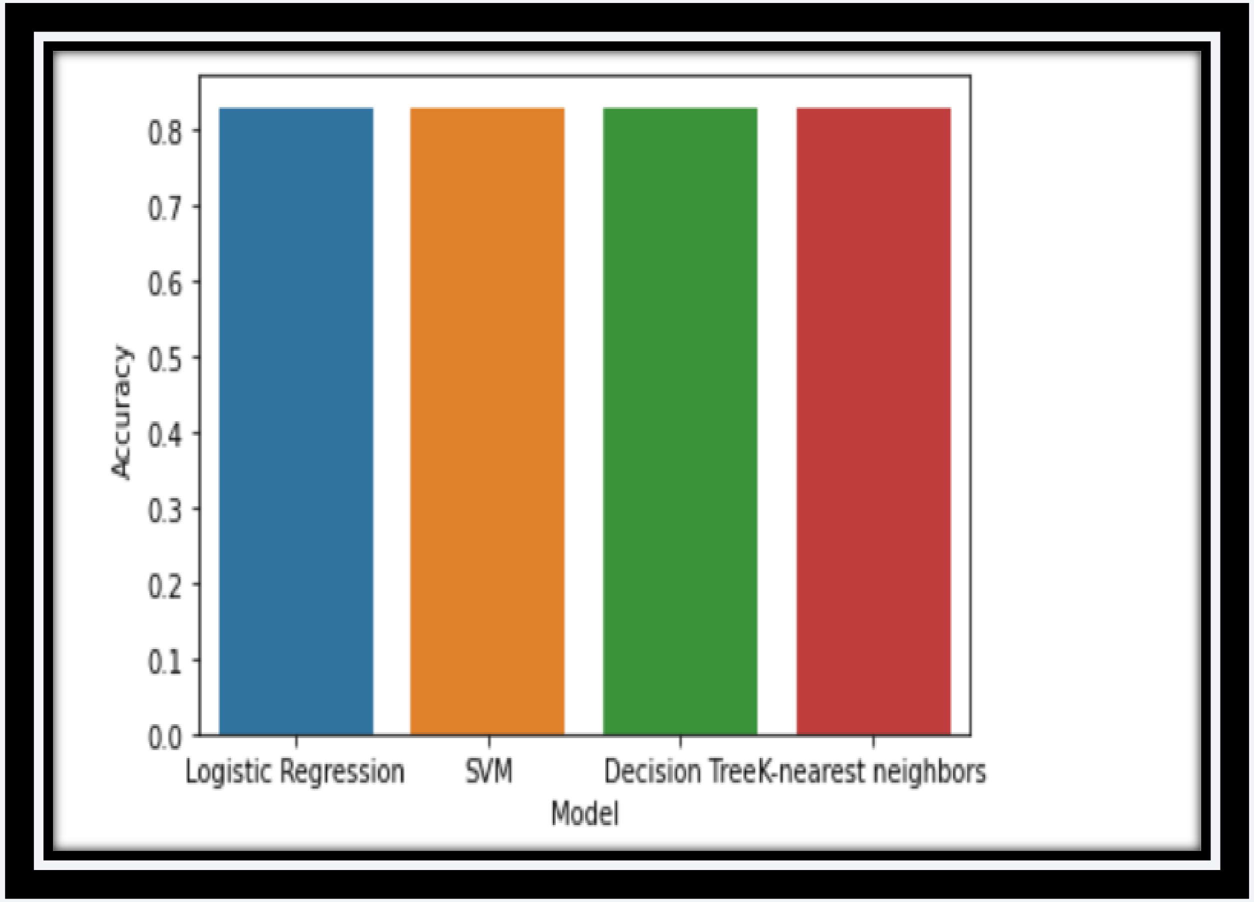


Finding: FT booster has the largest success rate. Most of the booster failed to land when the payload is important.

Section 6

Predictive Analysis (Classification)

Classification Accuracy

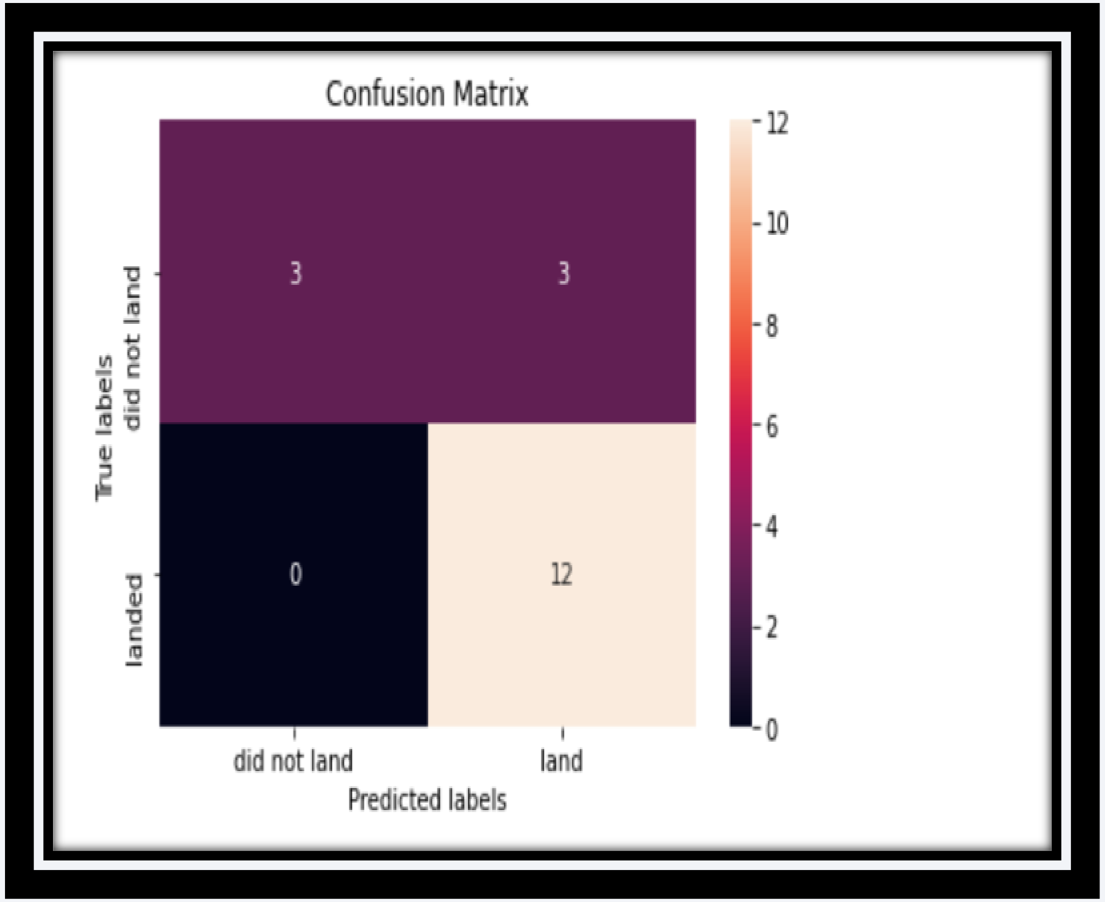


Finding: All the model's Logistic regression, SVM, Decision Tree, K-nearest Neighbors have the same test data accuracy.

Confusion Matrix

Finding:

- 12 records of successful landing were predicted correctly by the model.
- Zero false negative prediction.
- 3 records of failed landing were predicted correctly by the model.
- 3 records of failed landing were predicted as a successful landing.



Conclusions

- Most of launch sites are close to cost.
- The first stage will fail to land when Payload is important .
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- The success rate since 2013 kept increasing till 2020
- KSC LC-39 has the highest successful launches.
- All the models used in this project(Logistic regression, SVM, Decision Tree, K-nearest Neighbors) have the same test data accuracy.

Appendix

Code snippet: Creating bar chart for classification accuracy

```
In [28]: param={'Model':['Logistic Regression','SVM','Decision Tree','K-nearest neighbors'],'Accuracy':[0.83, 0.83,0.83,0.83]}
         param

Out[28]: {'Model': ['Logistic Regression',
           'SVM',
           'Decision Tree',
           'K-nearest neighbors'],
          'Accuracy': [0.83, 0.83, 0.83, 0.83]}

In [30]: dd=pd.DataFrame(param)
         dd

Out[30]:
```

	Model	Accuracy
0	Logistic Regression	0.83
1	SVM	0.83
2	Decision Tree	0.83
3	K-nearest neighbors	0.83

```
In [31]: sns.barplot(x="Model", y="Accuracy", data=dd)
```

Thank you!