IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<David Sumino>
<2022/02/09>

Outline

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

Executive Summary

- Methodology Summary
  - Analysis was done by gathering data through API and web scraping, important factors were extracted via data wrangling, and EDA was performed. EDA consisted of SQL queries, visualization through several libraries including scikit-learn, plotly, dash, and folium. Finally, several machine learning models were utilized and compared.

- Conclusions summary
  - Among the many factors observed, it seems that launch site, payload mass and orbit had significant impact on the final outcome. Overall, it seems that as time passed, the success rate of launches improved on the whole. Machine learning models could predict success with an accuracy of 88.8%.

All code can be found here:

https://github.com/shinobinomono/Data-Science-Capstone/tree/master

Introduction

- SpaceX is an American aerospace manufacturer. The advertised launch cost is $62 million USD for the Falcon 9 rocket. Much of the reason that the company can offer this price is because the initial phase of the launch can be recovered.
- The purpose of this analysis is to:
    - Determine the probability of recovering the initial phase of the launch
    - Determine what factors influence the success or failure of each launch
    - Map the launch sites to get insights and provide visual representation
    - Evaluation of machine learning models to predict future launch results

Section 1

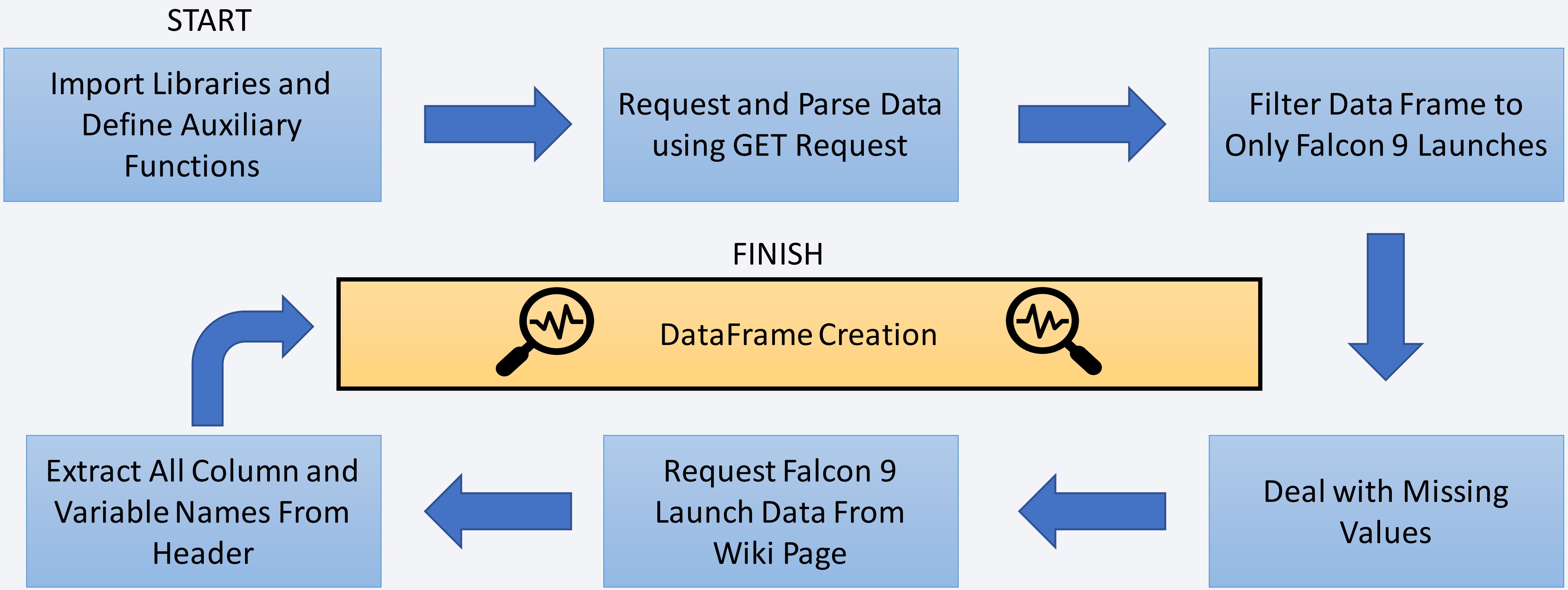
Methodology

Methodology

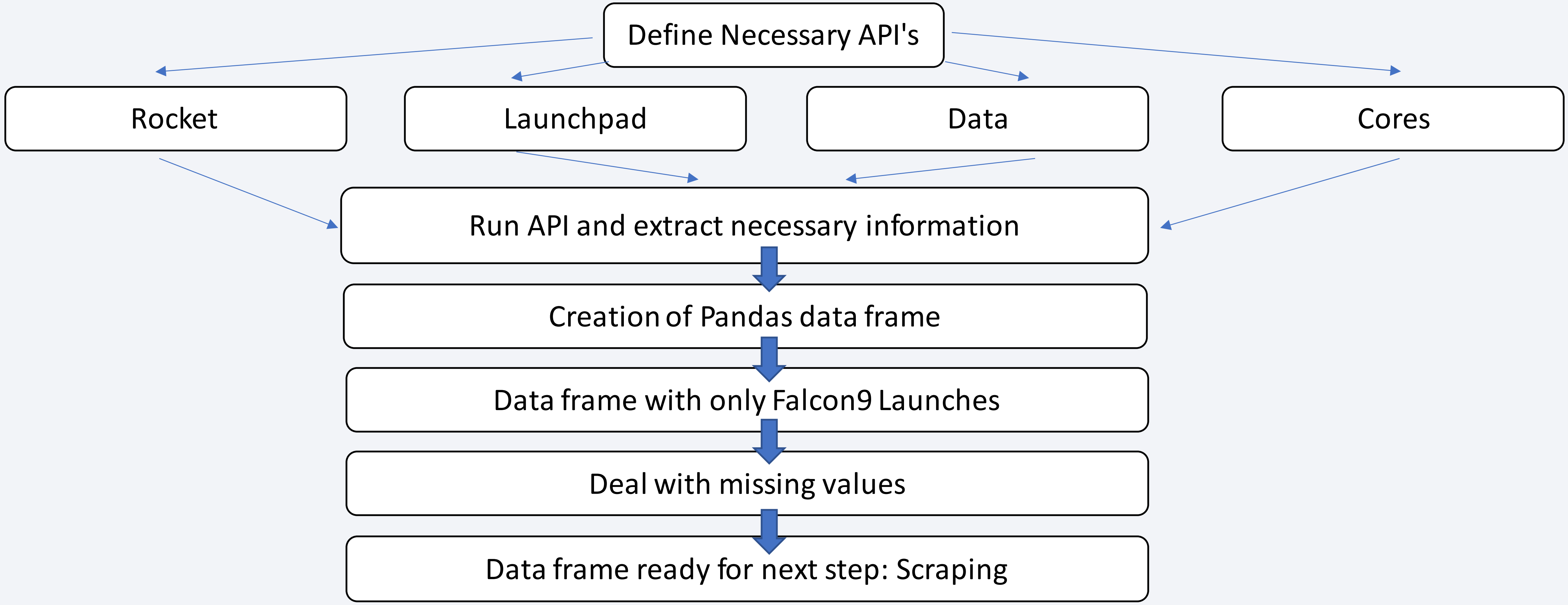
Executive Summary

- Data collection methodology:
    - Data was collected through SpaceX API, as well as web scraping of launch records found on Wikipedia
- Perform data wrangling
    - Data was wrangled using Pandas and NumPy. Important variables were brought into a new dataframe, and a new column was created to classify launch as Success of Failure
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Four classification models were compared in scikit-learn

Data Collection



Data Collection – SpaceX API



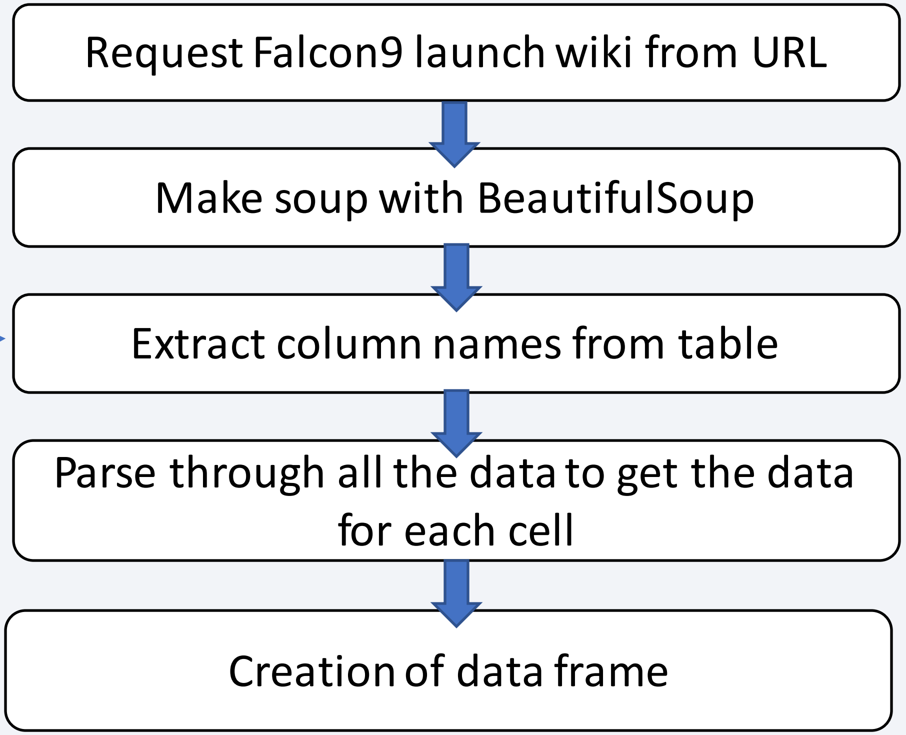
https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Data%20API%20Lab.ipynb

Data Collection - Scraping

Several Functions were used to find and extract the information

For example:
Which table
Header values
Table values

Finally, a 121 row x 11 column data frame was made by converting the dictionary of parsed information



https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Web%20Scraping%20Lab.ipynb

Data Wrangling

- Data set was loaded into pandas, and several exploratory processes were run.
- The number of launches, the number of each kind of orbit in the launch, and success criteria were identified.
- Through this analysis, a new key binary variable was produced - 'Class'
- This 'Class' variable represents the success or failure of the launch.

Load df → EDA → Identification of Outcomes → Creation of 'Class' variable

https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Data%20Wrangling%20Lab.ipynb

EDA with Data Visualization

- We will see some examples of the EDA with Visualization a bit later in the presentation
- Visualization includes numerous scatterplots, bar graphs, pie charts, and confusion matrix with heat map.

•Scatterplot: get a general idea of relationships

•Bar graph: see a side by side comparison with good detail of numbers

•Pie chart: offers a rough and impactful visual for variables with few categories

•Confusion matrix: offer visual comparison of accuracy of machine learning model prediction

https://github.com/shinobinomono/Data-Science-Capstone/blob/master/EDA%20with%20Visualization.ipynb

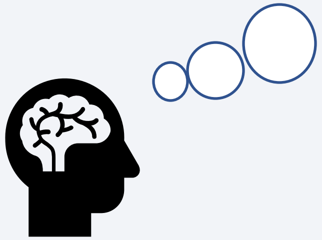
EDA with SQL

- We will look at the SQL queries in detail a bit further in the presentation

Examples of SQL statements used:

MIN, MAX
SUM
AVG
LIMIT
WILDCARDS

<u>Techniques</u>
Subquery
Operators (> < = !=)
Booleans (TRUE, FALSE)
Logicals (AND, OR)



https://github.com/shinobinomono/Data-Science-Capstone/blob/master/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- Markers and Circles were added for each of the launch sites in order to show location on the map
- Markers for each launch were added to the map, and colored to indicate whether the launch was considered a success (green), or failure (red)
- MarkerCluster was added to group together the icons who shared the same launch sites
- Finally, icons to show nearest coastline, rail line, highway, and airports were added, with lines to show the distances. These distances should be considered before launch to make sure that safety can be secured

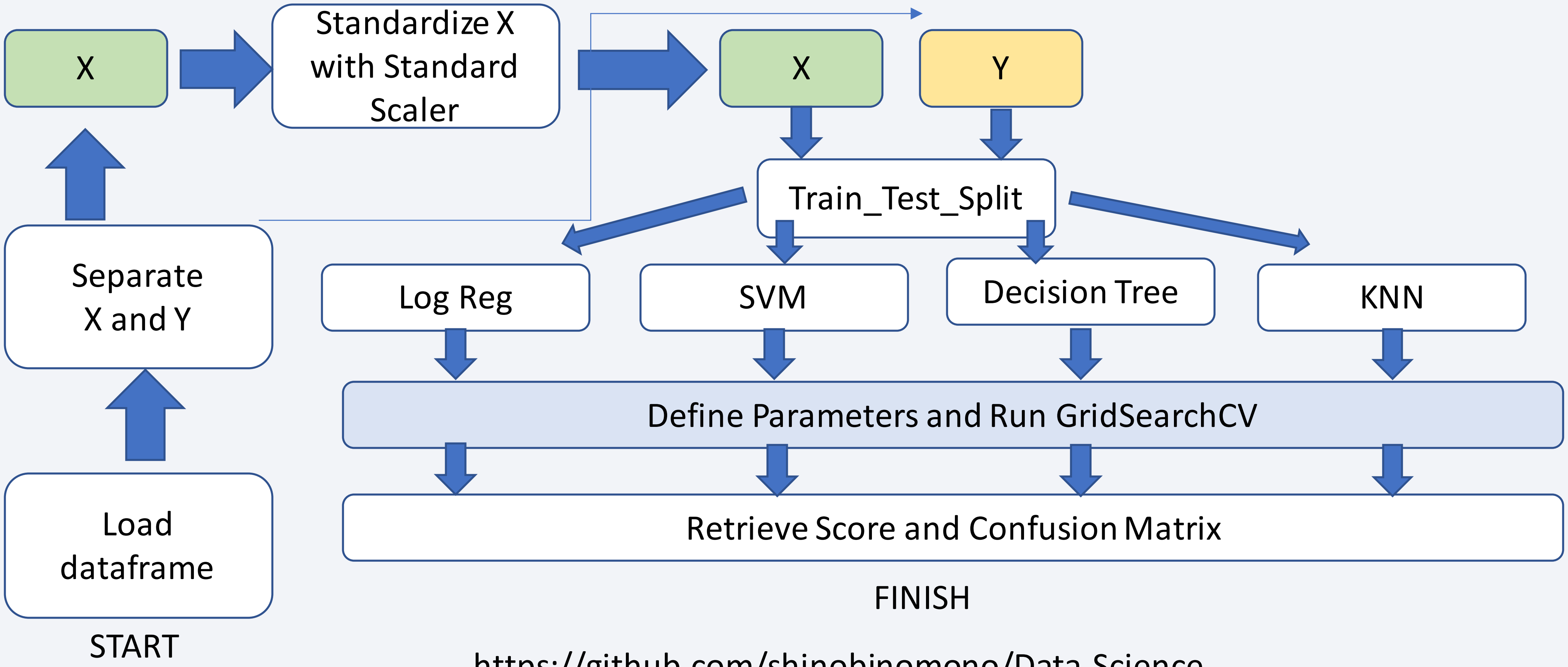
https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Interactive%20Visual%20Analytics%20-%20Folium.ipynb

Build a Dashboard with Plotly Dash

- The Dashboard with Plotly Dash consists of 2 main parts:
- **Pie chart**
  - Pie chart showing launches from each site, as well as success rates for each site
  - Pie chart is controlled with dropdown menu so you can choose from all sites or 1 site
- **Scatter Plot**
  - Scatter plot shows success and failure for each launch, plotted against payload
  - Scatter plot payload can be adjusted with a slide bar
  - Each booster is assigned a color to quickly distinguish which booster was used

https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Plotly%20Project%20(capstone).ipynb

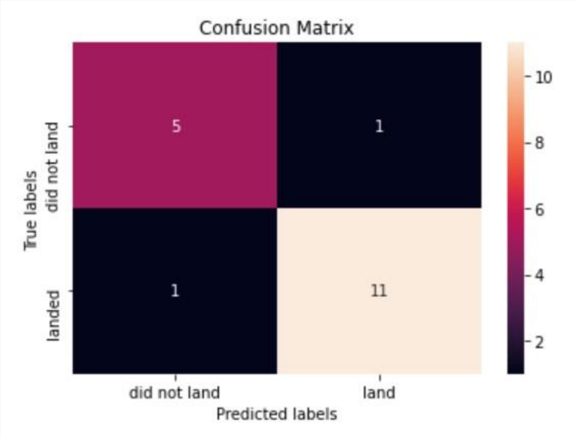
Predictive Analysis (Classification)

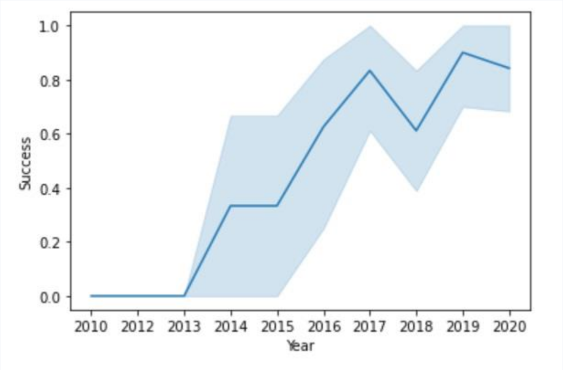


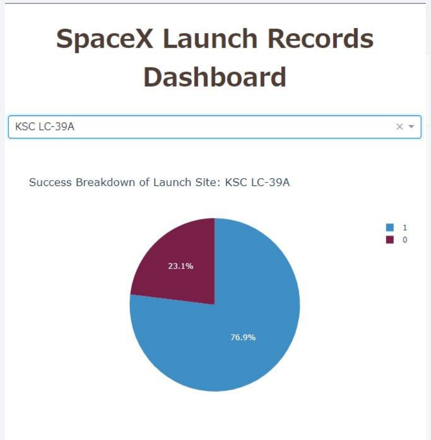
https://github.com/shinobinomono/Data-Science-Capstone/blob/master/Machine%20Learning%20Model%20Comparisons.ipynb

Results

- Exploratory data analysis showed that there were a few key factors that we should explore in more detail:
    - Payload mass, orbit, launch year, launch location







- We will see in more detail in later slides, but the best predictive model could predict the launch outcome with an accuracy of 88.8%

Section 2

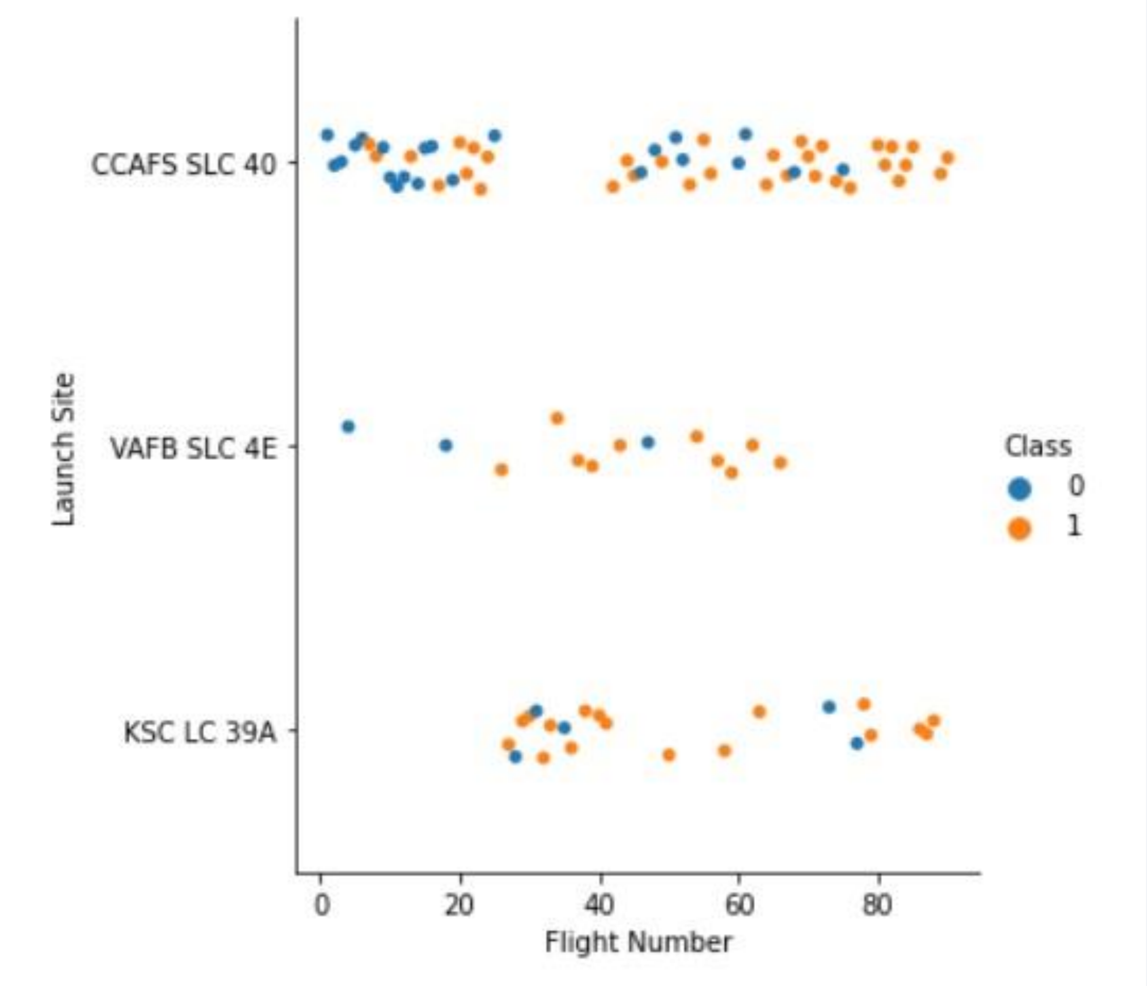
Insights drawn from EDA

Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site

Some observations:

- early launches experienced more failures
- there were concentrated launches at KSC LC 39A between Flight Number 20-40
- VAFB SLC 4E showed very good progress
- the majority of launches was at CCAFS SLC 40

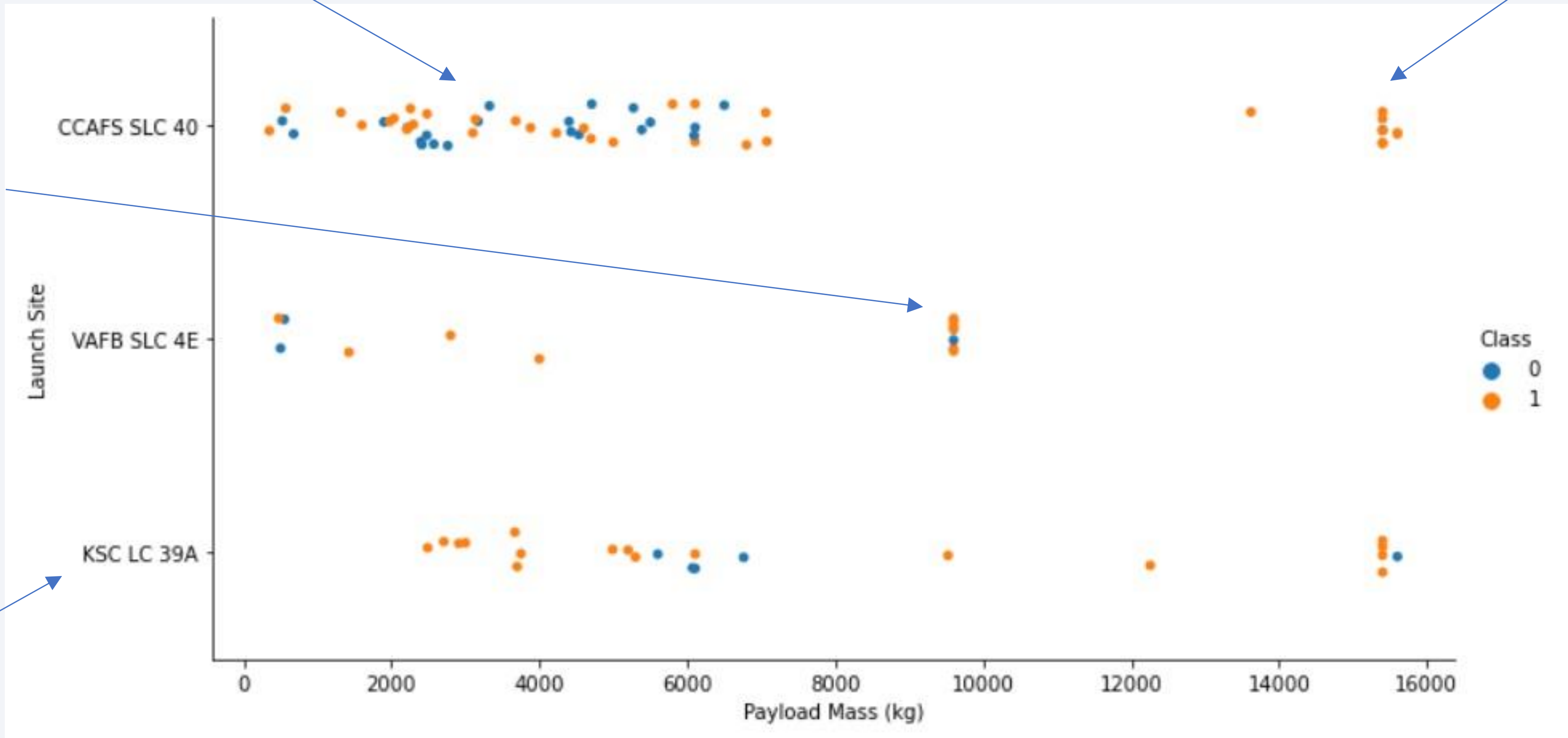


Payload vs. Launch Site

so-so

Nice!

VAFB SLC 4E has a relatively low max payload mass max



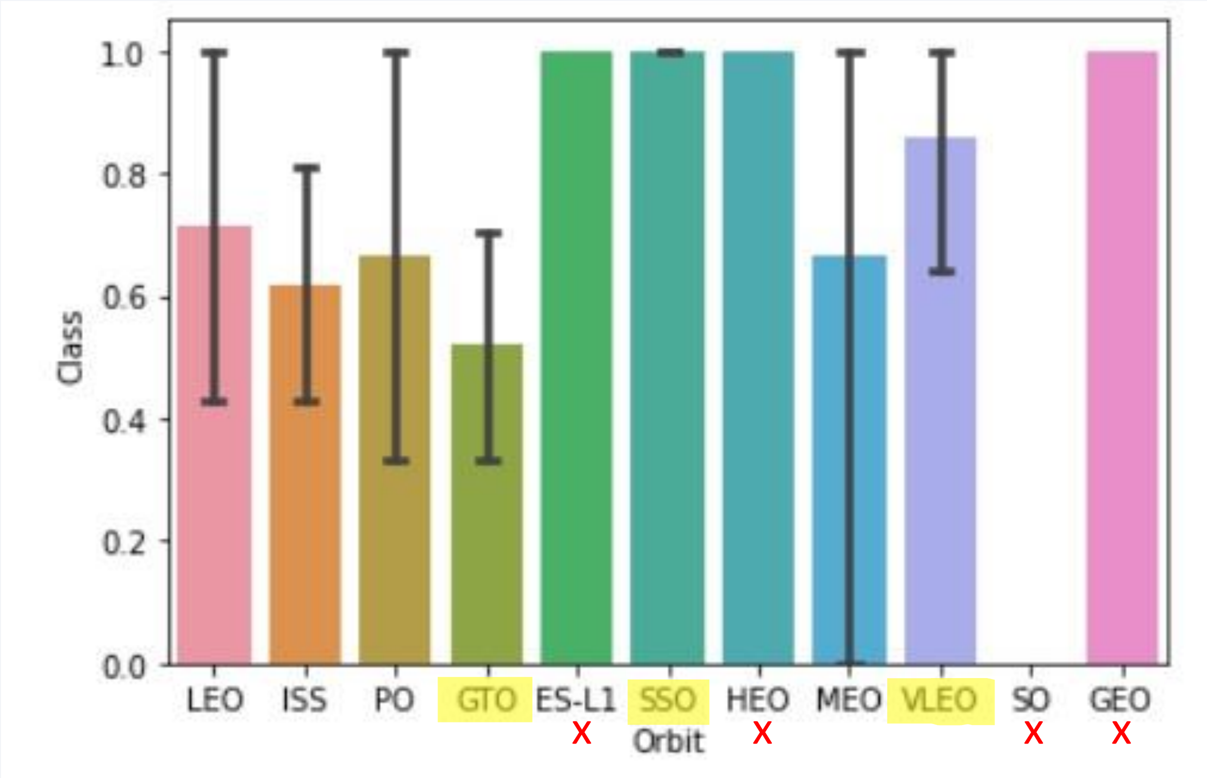
KSC LC 39A seems to have good results throughout the payload range

Success Rate vs. Orbit Type

Due to low data numbers, we cannot truly consider the orbits marked with a red x
(1 launch only)

Highlighted orbits are orbits of interest:

- GTO (13/27 success rate)
- SSO (5/5 success rate)
- VLEO (12/14 success rate)



Insights:

- **VLEO** seems to offer reliable success. However, VLEO was used in later launches
- **SSO** is promising, but needs more data to be sure
- **GTO** has the lowest success rate with most total launches. However, GTO was used in early launches.

*Details next slide

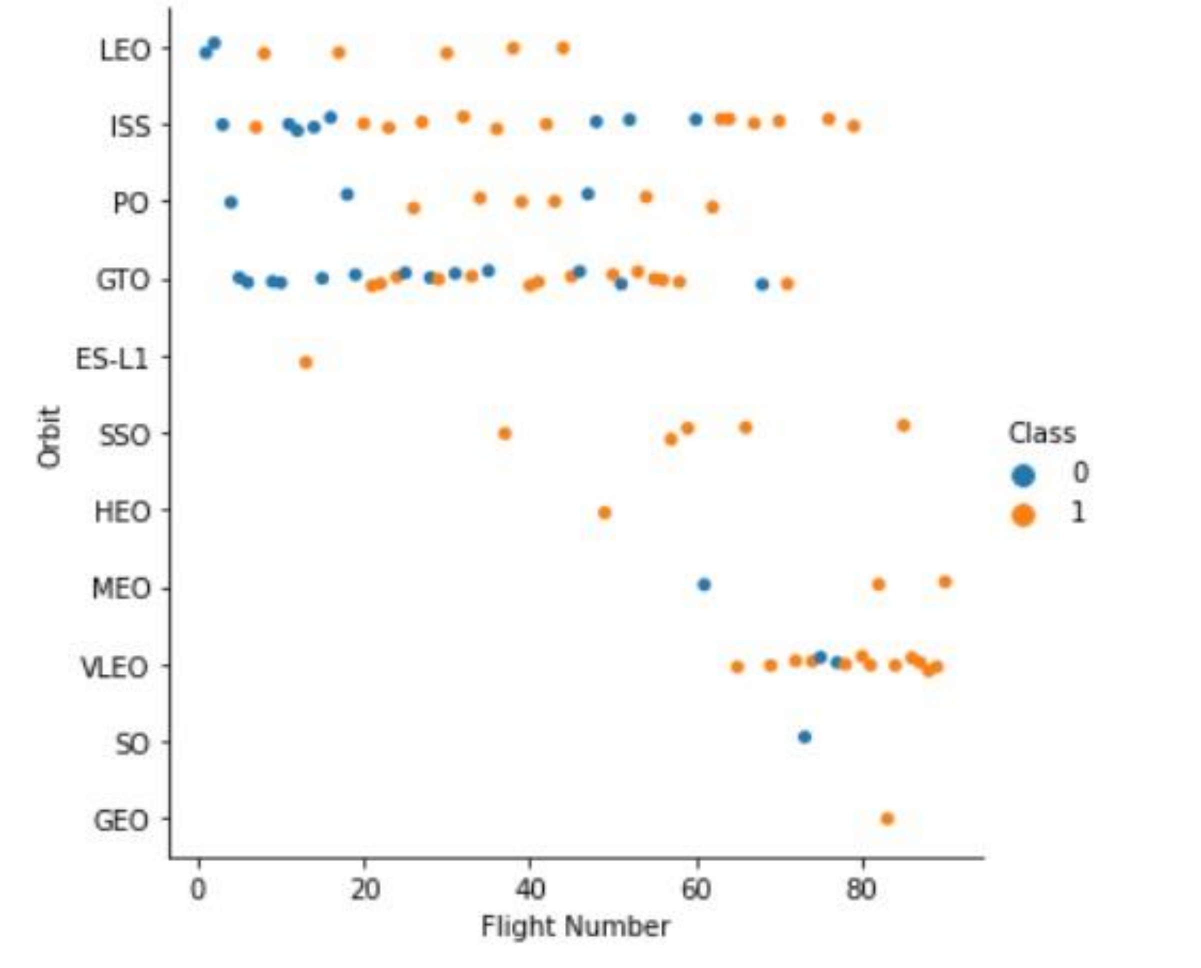
Flight Number vs. Orbit Type

VLEO has the highest reliable success rate as we saw in the previous slide.

However, we can see that it was used in later launches, which tend to have higher overall success rate.

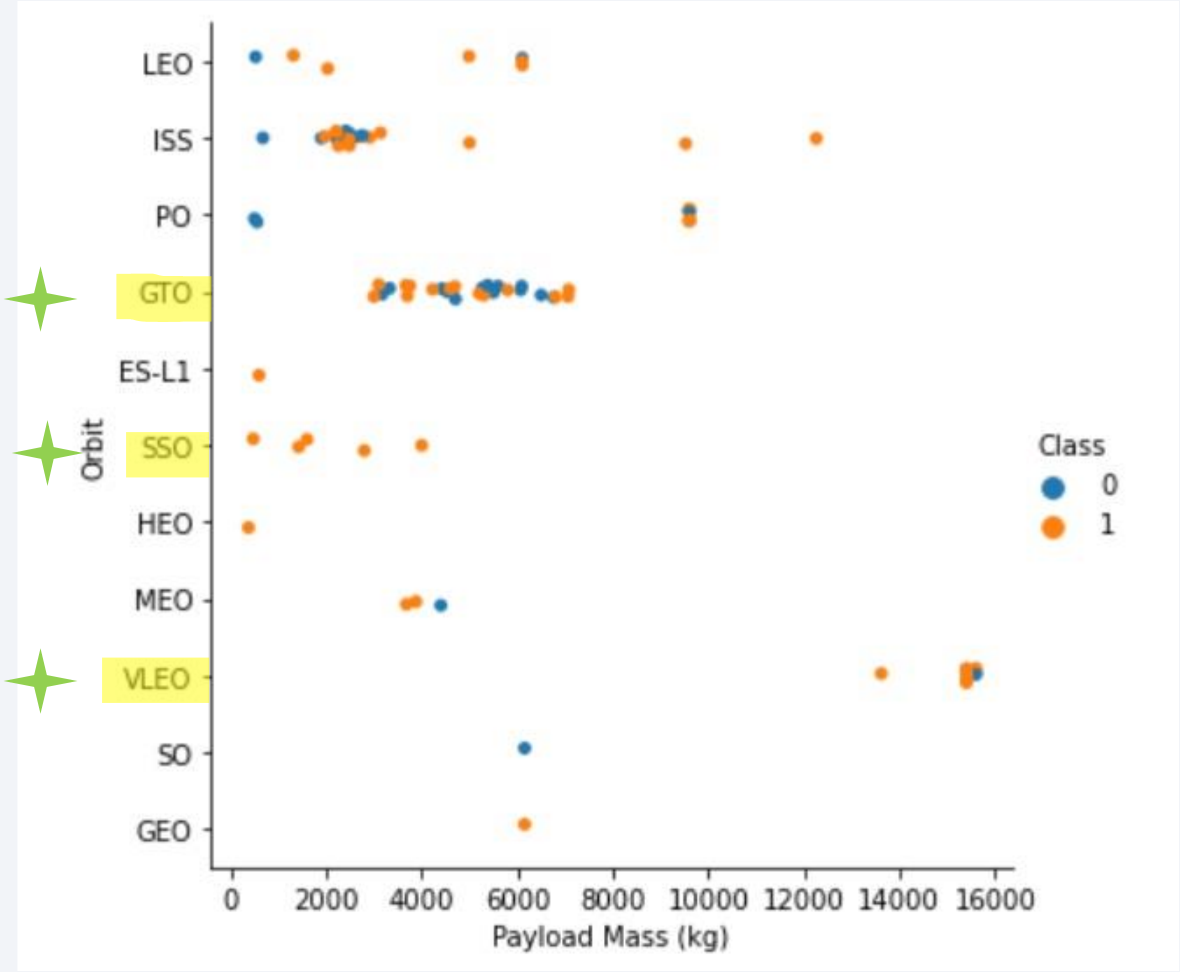
In contrast, GTO, which has the lowest overall success rate, seems to have been one of the first orbits.

After launch 38, it has nearly comparable success rate with VLEO (75% vs 86%).



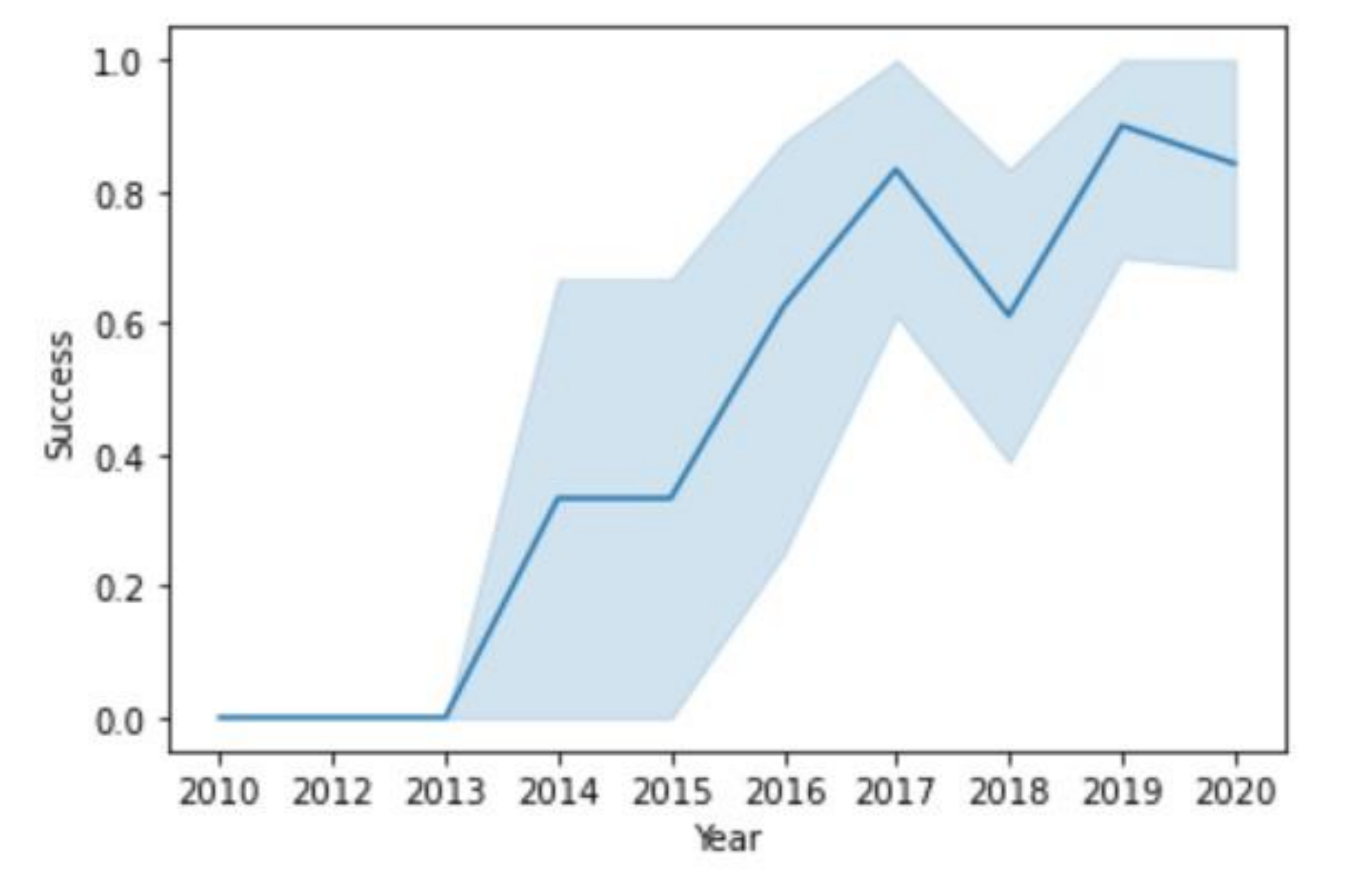
Payload vs. Orbit Type

- We can see a few key observations in this graph
- SSO has strong success, but only in small payload ranges
- GTO has a very concentrated payload range, and does not exceed 8000kg
- VLEO has a high kg payload average, as well as good success



Launch Success Yearly Trend

- We can see a clear trend towards successful launches
- 2014 was a big year with the first successful launches
- Success continued to increase until 2017
- There was a slight dip in 2018, followed by a peak in 2019



All Launch Site Names

Query was made to find all distinct launch sites.

We see that there are 4 launch sites in data set.

We use DISTINCT to make sure there is no overlap or doubled values.

In [38]:

```
%sql SELECT DISTINCT(launch_site) FROM SPACEXDATASET;
```

機密情報 TOP SECRET

Done.

Out[38]:

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

Launch Site Names Begin with 'CCA'

We wanted to check for launch sites that begin with 'CCA'.

Since we want to see the sites that BEGIN with 'CCA', the % mark is only at the end.

We add LIMIT 5 to keep the results to 5.

```
In [39]: %sql SELECT * FROM SPACEXDATASET ¥
         WHERE launch_site LIKE 'CCA%' ¥
         LIMIT 5;
```

機密情報 TOP SECRET

Done.

Out[39]:

launch_site	payload	payload_mass__kg_	orbit	customer	mission_out
CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Su
CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Su
CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Su
CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Su
CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Su

Note: In Japan, we use ￥ instead of ¥

Total Payload Mass

In [40]:
```
%sql SELECT SUM(payload_mass__kg_) FROM SPACEXDATASET ¥
WHERE customer = 'NASA (CRS)';
```

機密情報 TOP SECRET

Done.

Out[40]:

1
45596

Only when the customer is NASA

We want to find the total payload mass in KG for NASA.

We use SUM to find this answer.

It is 45,595kg in total.

Average Payload Mass by F9 v1.1

Only when the booster verson is F9 v1.1

```
In [41]: %sql SELECT AVG(payload_mass__kg_) FROM SPACEXDATASET ¥
         WHERE booster_version LIKE '%F9 v1.1%'
```

機密情報 TOP SECRET

Done.

Out[41]:

1
2534

We want to find the average payload mass in KG where the booster version is F9 v1.1

We use AVG to find this answer.

The average is 2,534 kg.

First Successful Ground Landing Date

```
In [42]: #come back... there is a problem with landing_outcome column
         #fixed... there was a mystery space in the landing_outcome column
         %sql SELECT MIN(launch_date) FROM SPACEXDATASET ¥
         WHERE landing_outcome LIKE '%uccess%'
```

機密情報 TOP SECRET

```
Done.

Out[42]:
```

1
01-05-2017

Query was run to find the earliest successful ground landing date.

MIN was used to find the earliest.

Successful Drone Ship Landing with Payload between 4000 and 6000

Used a combination of the previous techniques:

- DISTINCT
- WHERE >
- WHERE <
- LIKE %uccess%

In [8]:
```
%sql SELECT DISTINCT(booster_version) FROM SPACEXDATASET ¥
WHERE payload_mass__kg_ >4000 and payload_mass__kg_ ¥
<6000 and landing_outcome LIKE '%uccess';
```

機密情報 TOP SECRET

Done.

Out[8]:

booster_version
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5 B1058.2
F9 B5B1060.1
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

Query was run to find the number of successful and failure mission outcomes.

COUNT was used.

In [44]:
```
%sql SELECT count(*) as number_of_successes FROM SPACEXDATASET ¥
WHERE mission_outcome LIKE '%uccess%'
```

機密情報 TOP SECRET

Done.

Out[44]:

number_of_successes
100

In [45]:
```
%sql SELECT count(*) as number_of_failures FROM SPACEXDATASET ¥
WHERE mission_outcome LIKE '%ail%'
```

機密情報 TOP SECRET

Done.

Out[45]:

number_of_failures
1

Boosters Carried Maximum Payload

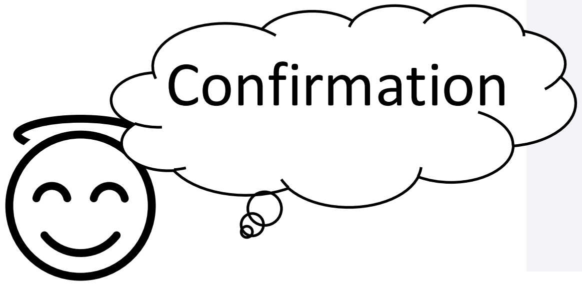
In [46]:
```
%sql SELECT DISTINCT(booster_version), payload_mass__kg_ FROM SPACEXDATASET ¥
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET)
```

機密情報 TOP SECRET

Done.

Out[46]:

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



In [47]:
```
#checking the answers
#%sql SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET
```

機密情報 TOP SECRET

Done.

Out[47]:

1
15600

Query to show a list of booster versions which carried maximum payload.

Used a subquery to get all the information selected

2015 Launch Records

```
In [48]: #should have worked, but could not extract year from the series
         #https://developer.ibm.com/articles/fun-with-dates-and-times/
         #%sql SELECT landing_outcome, booster_version, launch_site FROM SPACEXDATASET ¥
         #WHERE YEAR(DATE) = 2015

         %sql SELECT landing_outcome, booster_version, launch_site, launch_date FROM SPACEXDATASET ¥
         WHERE launch_date LIKE '%2015%' and landing_outcome LIKE '%ail%'
```

機密情報 TOP SECRET

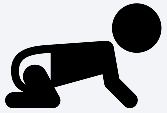
```
Done.
```

Out[48]:

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

Query was run to find the 2015 launches that had a failed outcome.

I had a lot of trouble extracting dates. So, I just used a wildcard.

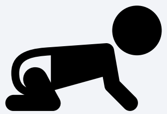


Probably not the best option, but it worked.

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

Query was run to rank the outcomes between these two dates.

Again, I had trouble extracting dates, so I just brute forced it.



Seems to have worked.

In [101]:

```
%sql SELECT landing_outcome, count(*) as count FROM SPACEXDATASET ¥
WHERE launch_date LIKE '%2010%' ¥
OR launch_date LIKE '%2011%' ¥
OR launch_date LIKE '%2012%' ¥
OR launch_date LIKE '%2013%' ¥
OR launch_date LIKE '%2014%' ¥
OR launch_date LIKE '%2015%' ¥
OR launch_date LIKE '%2016%' ¥
OR launch_date LIKE '%16-03-2017%' ¥
OR launch_date LIKE '%02-2017%' ¥
OR launch_date LIKE '%01-2017%' ¥
GROUP BY landing_outcome ¥
ORDER BY count(*) desc;
```

機密情報 TOP SECRET

Done.

Out[101]:

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

Section 3

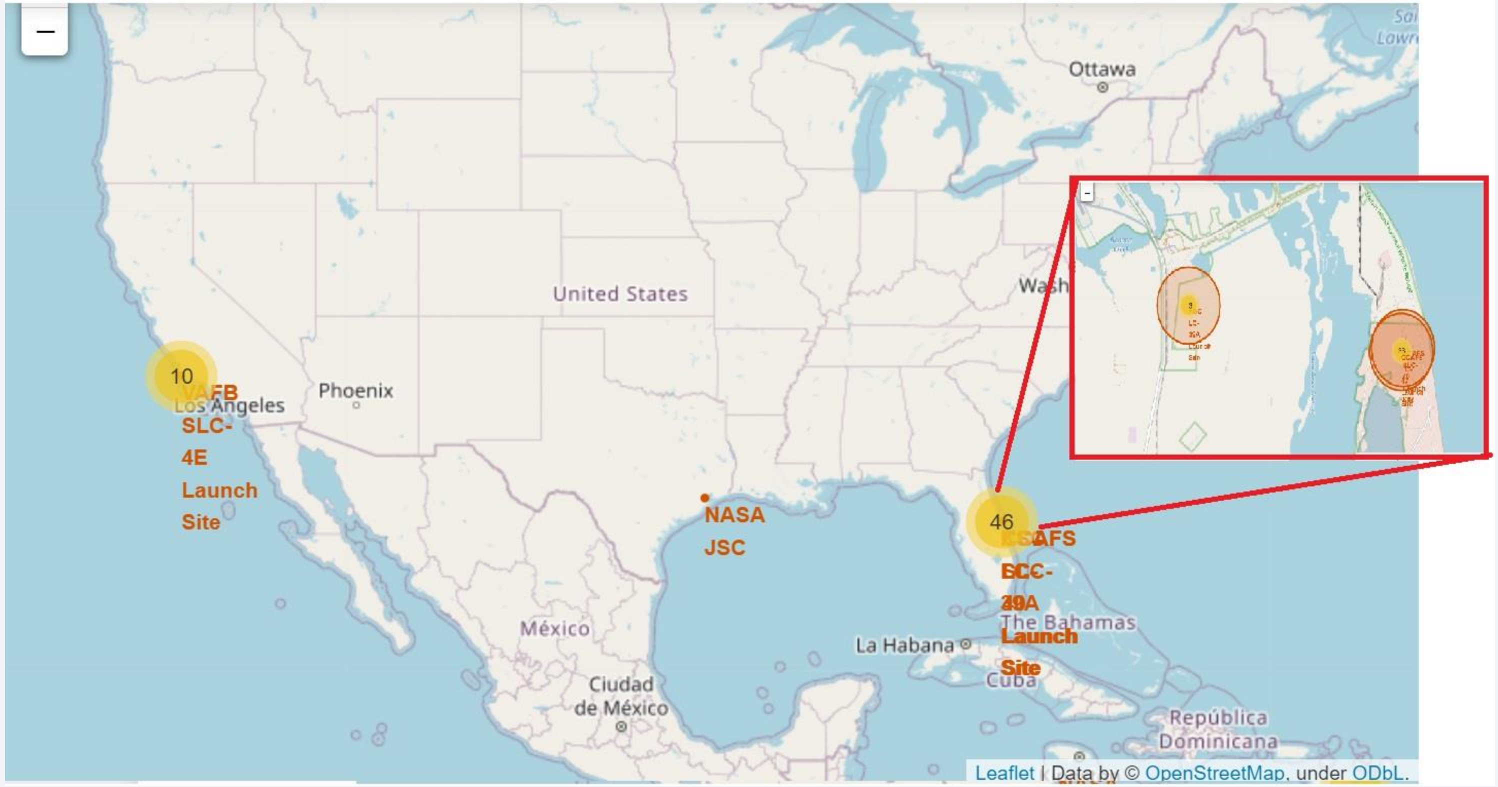
Launch Sites Proximities Analysis

Folium Map of Launch Sites

We can see from the Folium map that there are launch sites in 2 areas: California and Florida.

Florida actually has 3 sites.

2 of them are in VERY close proximity to each other, and the other is slightly West.



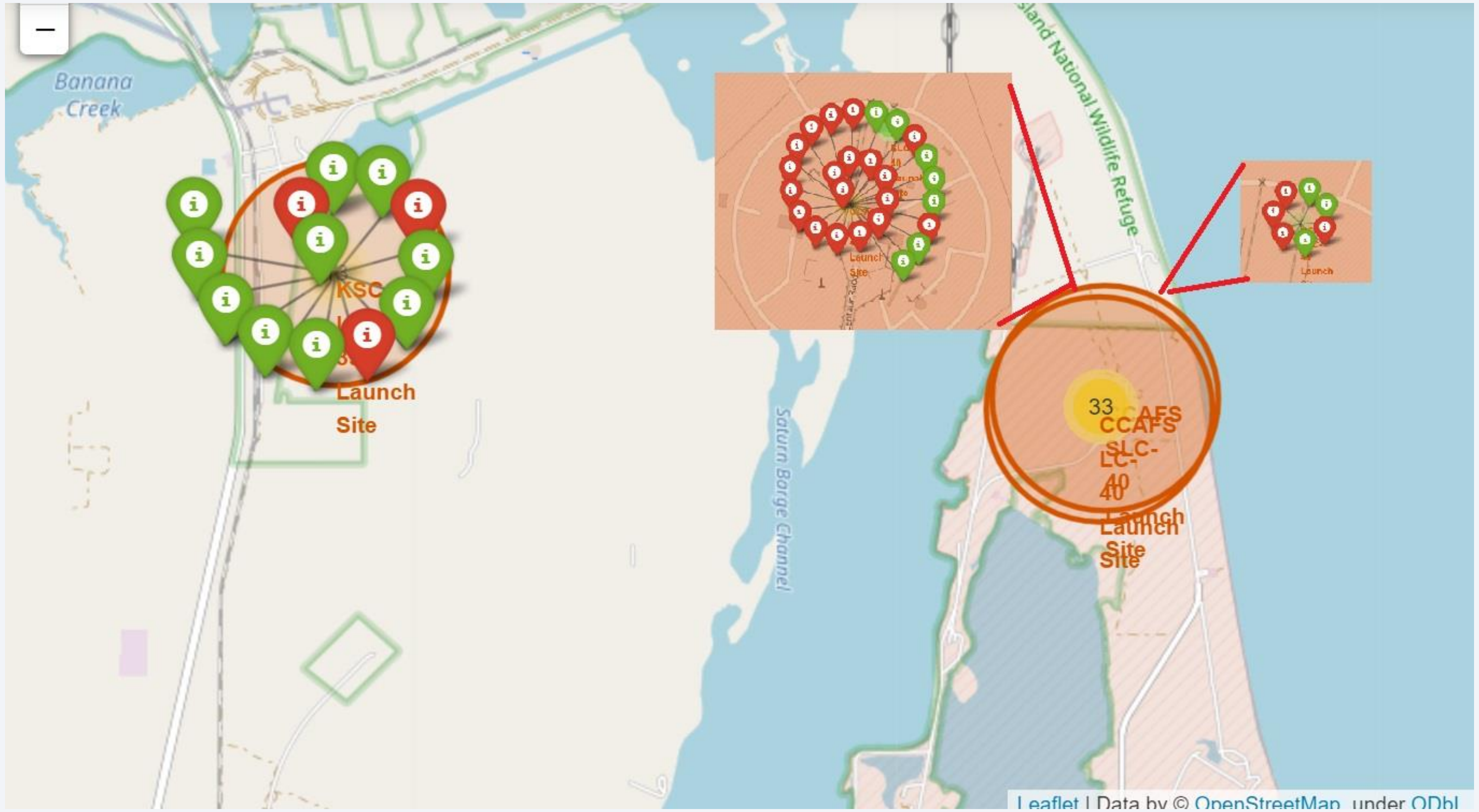
We can see this in the zoomed in area.

A Closer Look at the Florida Launch Sites

Here we can see the difference in launch sites.

The KSC LC-39A site (left) has lots of green, which means the success ratio is high.

In contrast, the CCAFS LC-40 and CCAFS SLC-40 sites (right) are considerably lower.



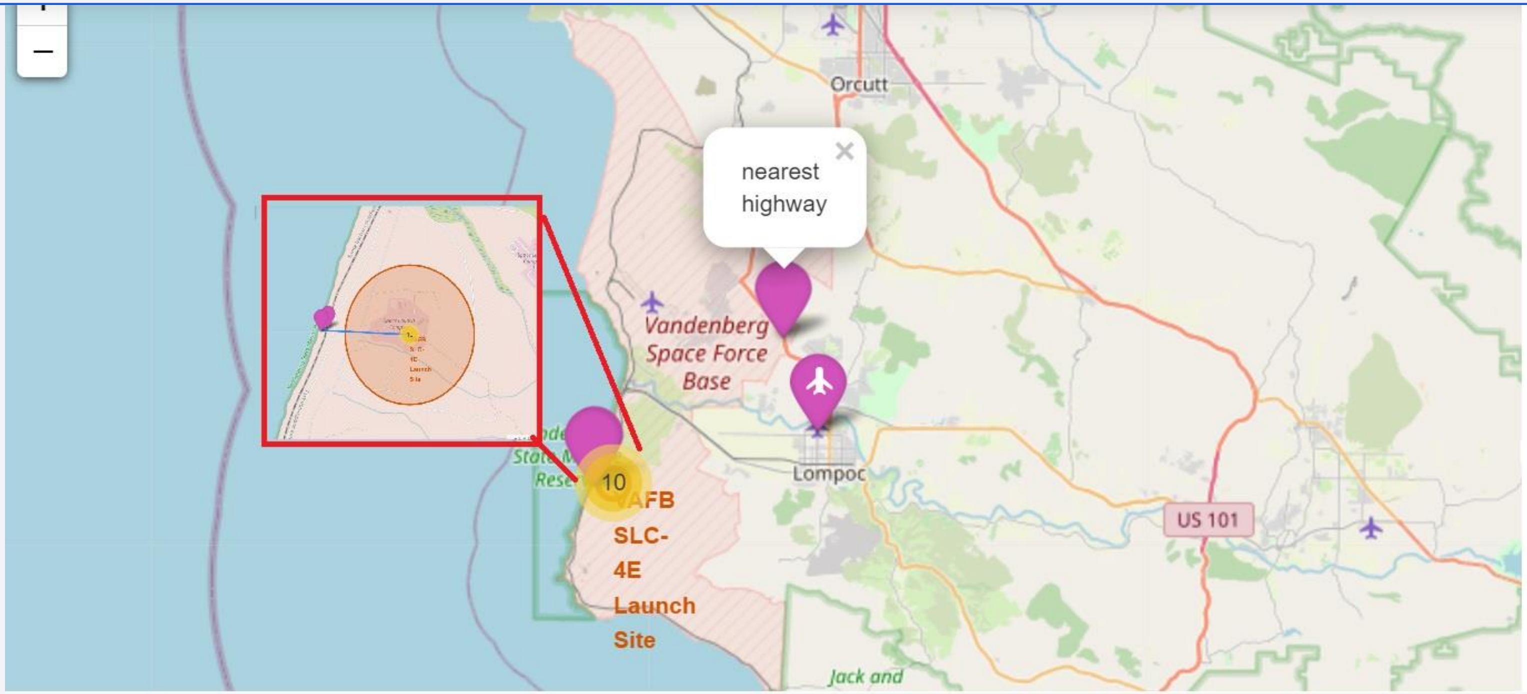
Green: Success
Red: Failure

California Launch Site: Distance From Stuff

Let's give the California launch site some love, too.

Coastline distance is important for ease of recovery of the first stage from the ocean drone ship.

The distances from highway, airport, and railway are important because you don't want explosions exploding and injuring people or infrastructure.



万川集海

```
distance_railway = calculate_distance(34.632834,-120.610746,34.63383,-120.62463)
distance_highway =calculate_distance(34.632834,-120.610746,34.71932,-120.49084)
distance_coastline = calculate_distance(34.632834,-120.610746,34.63339,-120.62578)
distance_airport = calculate_distance(34.632834,-120.610746,34.66484,-120.46680)
print("distance to railway:", round(distance_railway, 3), "km ¥n", "distance to highway:", round(distance_highway, 3), "km ¥n",
      "distance to coastline:", round(distance_coastline, 3), "km ¥n", "distance to airport:", round(distance_airport, 3), "km ¥n")
```

```
distance to railway: 1.275 km
 distance to highway: 14.589 km
 distance to coastline: 1.377 km
 distance to airport: 13.644 km
```

Section 4

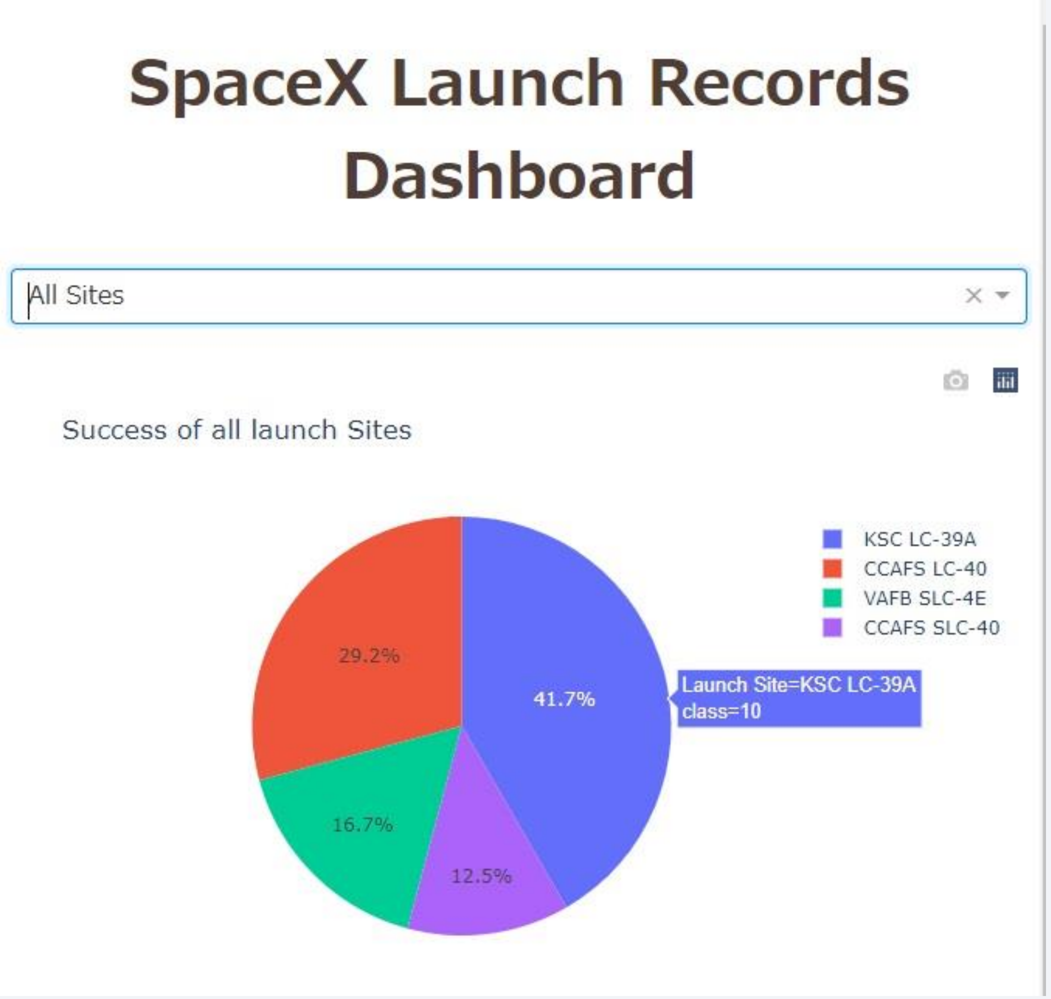
Build a Dashboard with Plotly Dash

Launch Success Count for All Sites

This screenshot of the dashboard shows a breakdown of all 4 launch sites.

As we can see, KSC LC-39A has the largest number of launches.

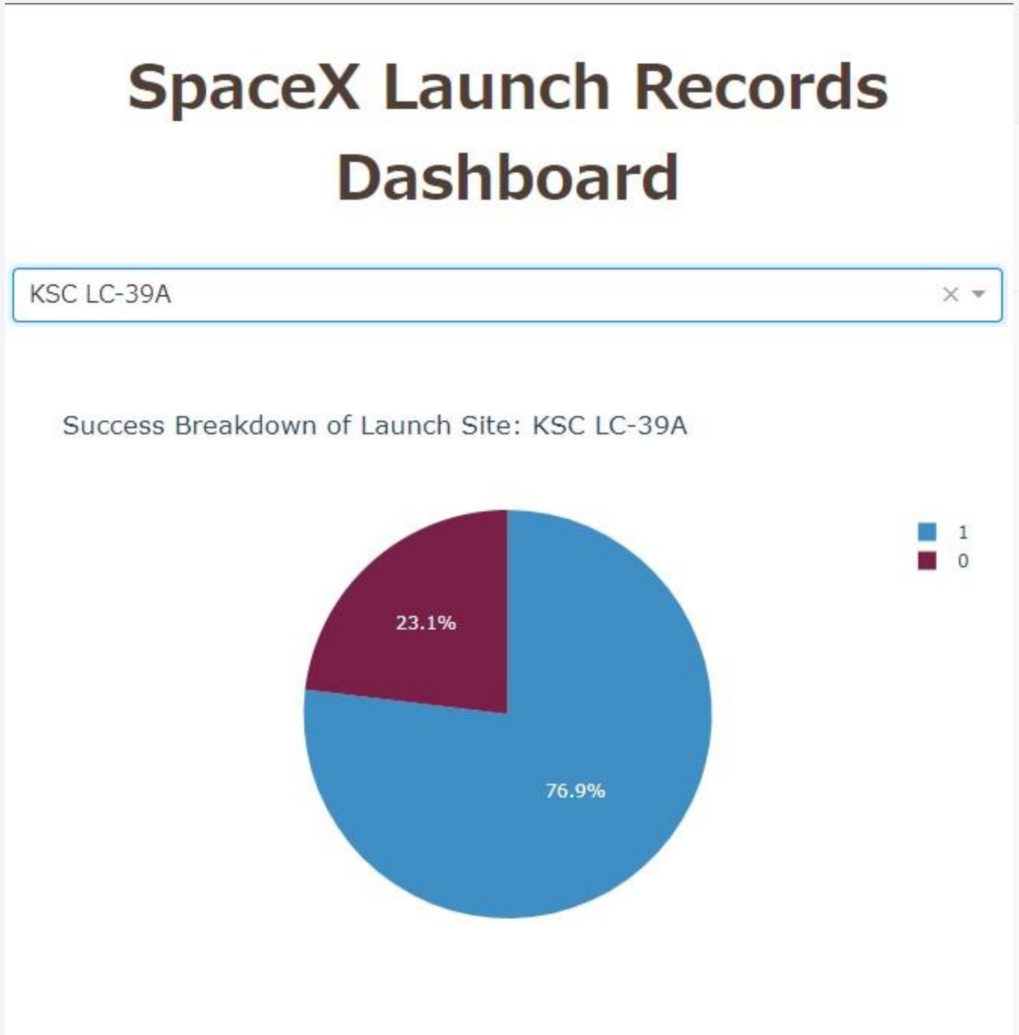
Incidentally, it also has the highest success rate.



Highest Launch Success Ratio: KSC LC-39A Site

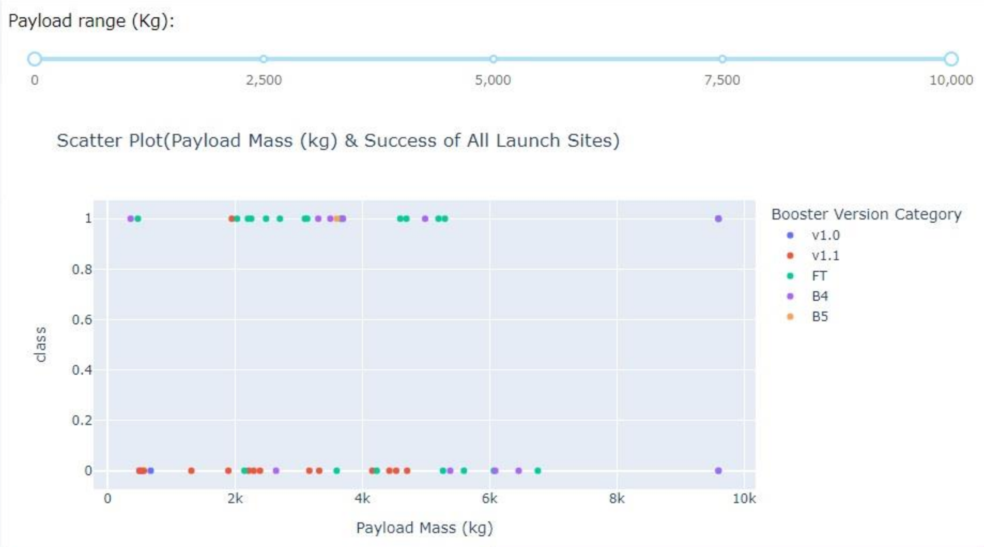
KSC LC-39A gives us the highest success rate.

The success rate is 76.9%



Scatter Plot: Payload vs. Launch Outcome

Launch Outcomes
for All Payloads



Let's look into the range of 2000-6000kg.

Here we can see that we have considerably better success ratio.

Notice that FT Booster Version does very well in this range.

We can also see that V1.1 Booster version is the least successful.

Section 5

Predictive Analysis (Classification)

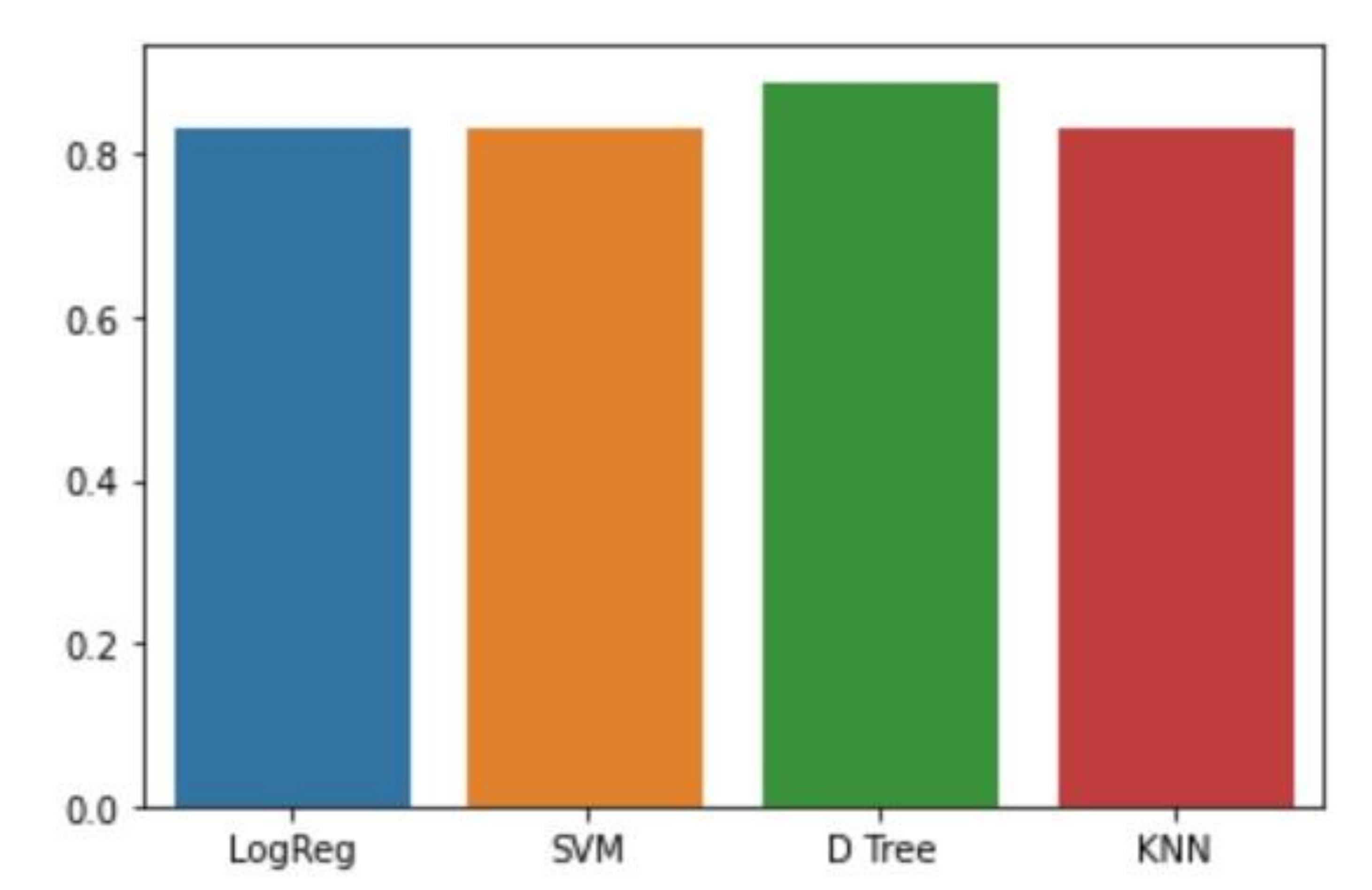
Classification Accuracy

Logistic Regression
-83.3%

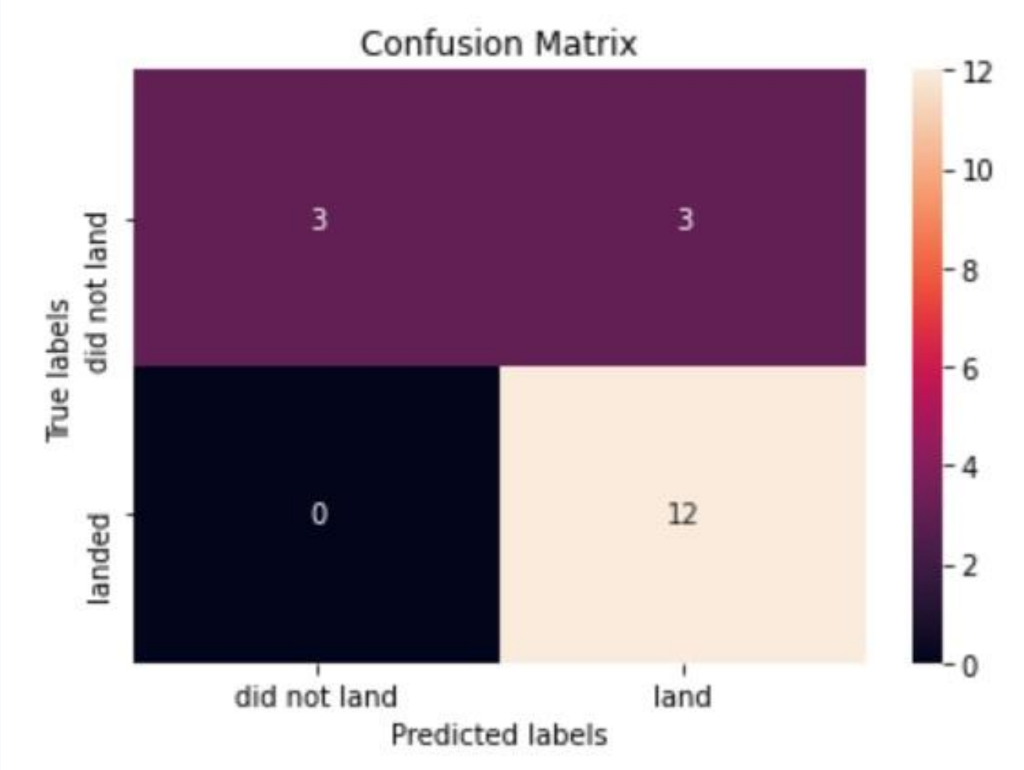
Support Vector Machine
-83.3%

Decision Tree
-88.8%

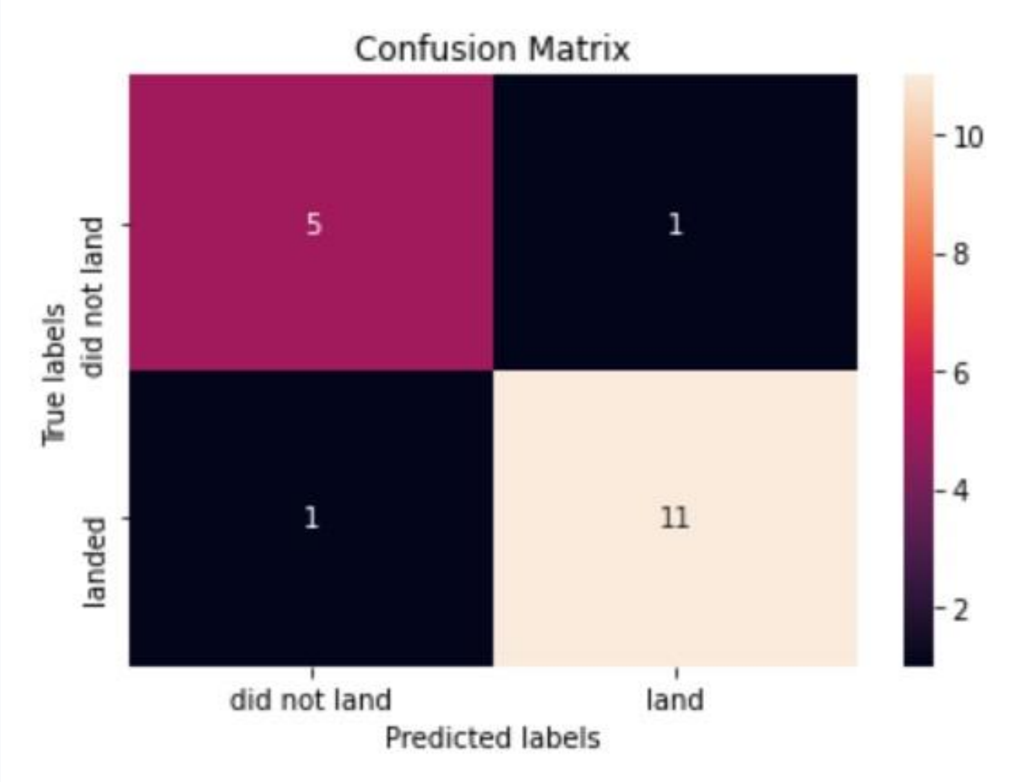
K Nearest Neighbors
-83.3%



Confusion Matrix



KNN, SVM, Logistic Regression



Decision Tree

Let's take a look at our most accurate model, the **decision tree.**

It was slightly more accurate than the other models, and had one less error in total.

Conclusions

- Success rose significantly over time
- Success rate varied with orbit
  - Ranged from 48.1%(GTO) to 85.7%(VLEO)
- Success rate varied with payload mass
  - Sweet spot between 2000kg and 6000kg
- Success rate varied by launch site
  - KSC LC-39A was the most successful with 76.9% success rate
- Machine learning models could predict launch success/failure
  - Decision tree could predict with 88.8% accuracy

Appendix

Finding the best parameters for the model

In [79]:
```
parameters3 = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

treez = DecisionTreeClassifier()
```

In [80]:
```
tree_cv = GridSearchCV(treez, parameters3, cv=10)
```

In [81]:
```
tree_cv.fit(X_train, Y_train)
```

Out[81]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In [82]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_
samples_split': 10, 'splitter': 'random'}
accuracy : 0.8892857142857145
```

Thank you!