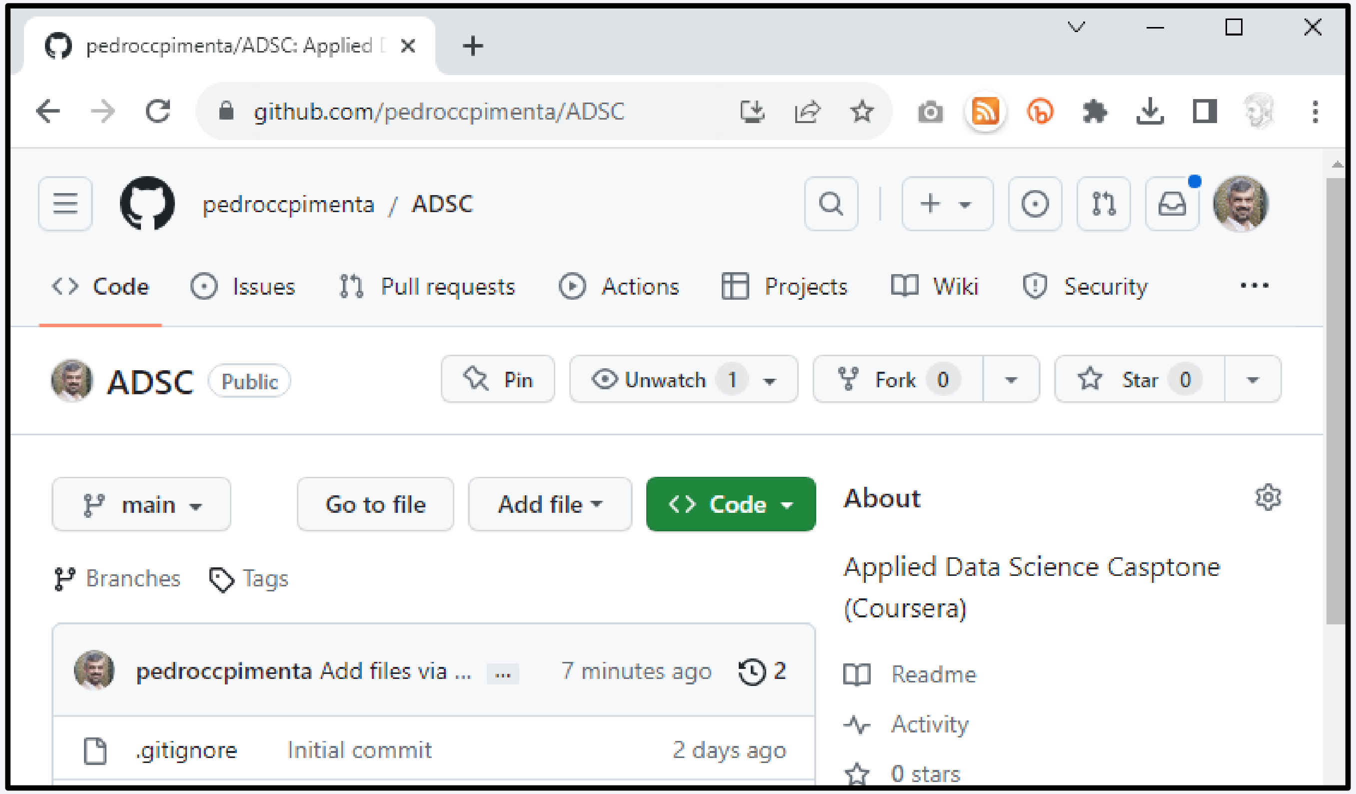
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Pedro Pimenta
2023-08-16

Outline

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



https://github.com/pedroccpimenta/ADSC

Executive Summary

- Summary of methodologies
  - In this report we present the full data analysis of Space-9 launching data as prescribed in the IBM / Coursera "Applied Data Science Capstone" Course – In this analysis:
    - we used both a REST API and webscrapping methods to get relevant data for the present analysis;
    - we made same preliminary analysis with direct SQL, chart and map visualization;
    - we produced an interactive dashboard, thus allowing final users to perform their own analysis;
    - and we applied a set of ML algorithms to predict the success of recovering the first stage of the rocket
- Summary of all results
  - Data harvesting methods guarantee we easily have data updated
  - Data available needs some cleaning before final analysis
  - A interactive dashboard was setup in order to promote further input from the field experts
  - Prediction is possible with a score of - further analysis would be required
  - Methods used are aligned with purpose and context, although further research is advised

Introduction

- Project background and context
  - SpaceX promotes Falcon 9 rocket launches on its official website at a price point of (only) $62 million. In contrast, alternative providers command prices exceeding $165 million per launch. A significant portion of this cost disparity stems from SpaceX's unique ability to recover and reuse the initial stage of the rocket. As a result, the feasibility of predicting the successful landing of the first stage becomes pivotal in estimating the overall launch expenses..
- Problems you want to find answers
  - Our job is to demonstrate how some methods from “Data Science” can be setup to predict, based on past, public data, if the first stage of a given launch will be recovered or not, thus predicting the launch costs, by:
  - Identifying key factors in first stage landing success / failure;
  - Establishing score for predictions with available data

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected through a REST API from SpaceX and webscrapping a Wikipedia page devoted to SpaceX launches. We use interactive, manual Jupyter Notebooks (and thus the process can be easily automated through Apache Airflow / Kubeflow, etc, if considered necessary)

Methodology

- Data wrangling
  - Data is of relative high quality, and just minor cleaning (removing white lines and replacing a few missing values) was necessary – pandas methods were use;
  - Through standard requests / json / BeautifoulSoup we could get the dataframes necessary for further analysis.
  - This dataframes are stored in csv files

Methodology

- Perform exploratory data analysis (EDA) using visualization and SQL
  - Dataframes obtained in previous steps are ready to be directly queried through xSQL interfaces and charted / visualized with matplotlib, seaborn, etc
  - Some descriptive statistics and simple charts were obtained

Methodology

- Perform interactive visual analytics using Folium and Plotly Dash
  - Considering the geographical nature of the data, we used Folium to gain some insights about the launching locations (further analysis is advised regarding landing locations)
  - An interactive Dash board (Dash + plotly) was setup to easy the involvement of field experts in the analysis

Methodology

- Perform predictive analysis using classification models
  - Eventually, some machine learning algorithms were trained over the available data (KNN, SVM, Logistic Regresion, Support Vectora and Decision trees), and some predictions obtained. Sklearn library was used.

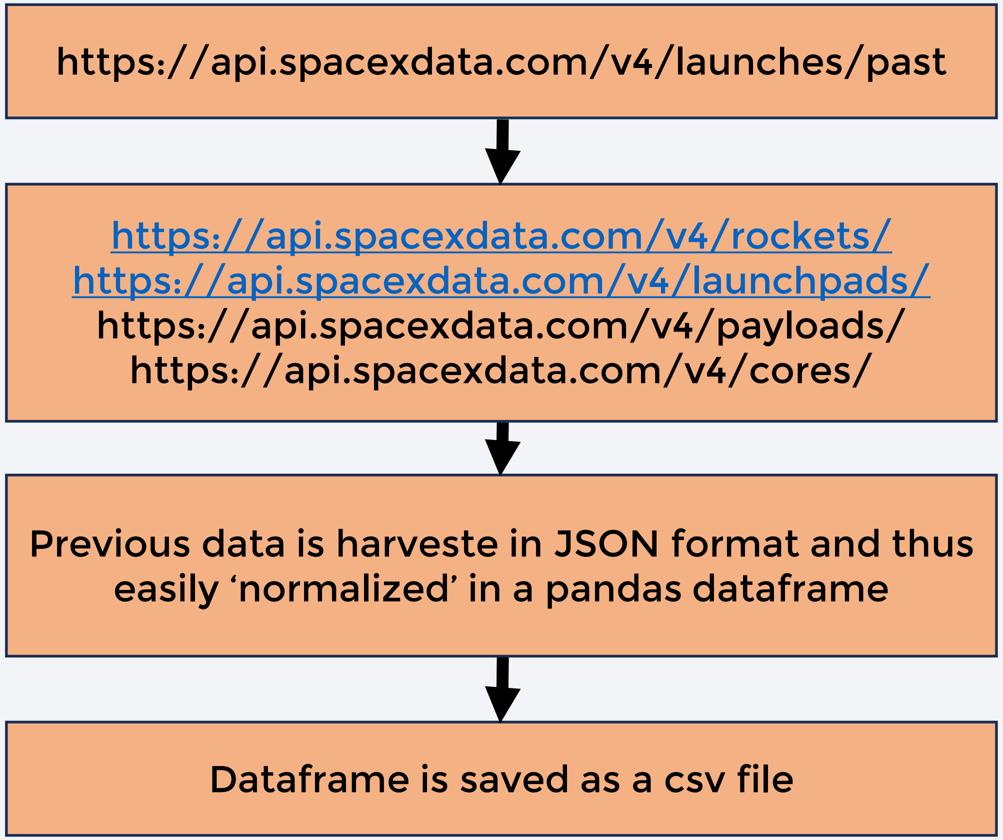
Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

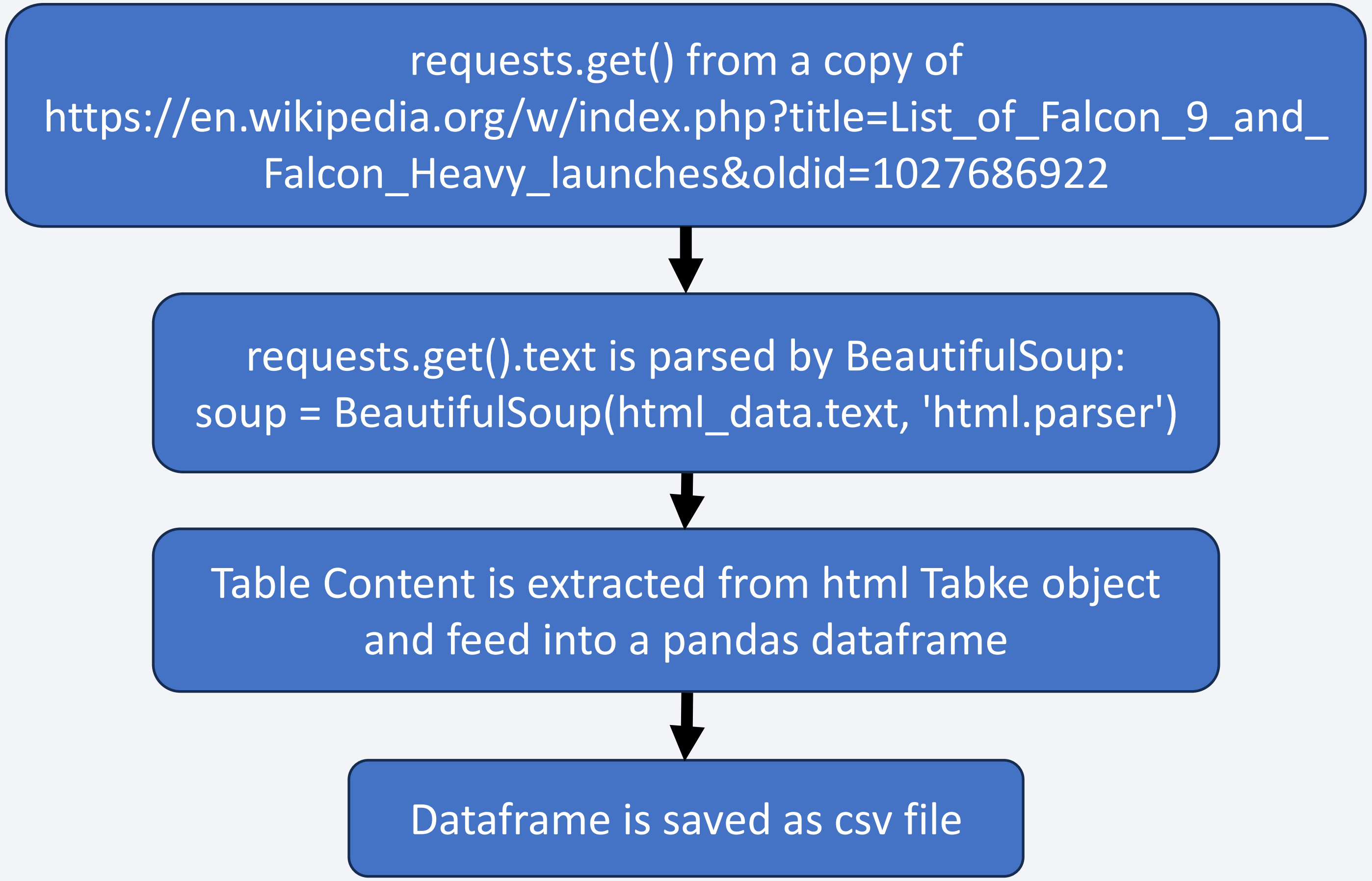
- First set of data regards “past launches” (end point /launches/past) – next we get more data from the endpoints ‘rockets’, ‘launchpads’, ‘payloads’, and ‘cores’ -> JSON data is ‘normalized’ to a pandas dataframe -> dataframe is saved as a csv file

- Full Jupyter Notebook is available here.



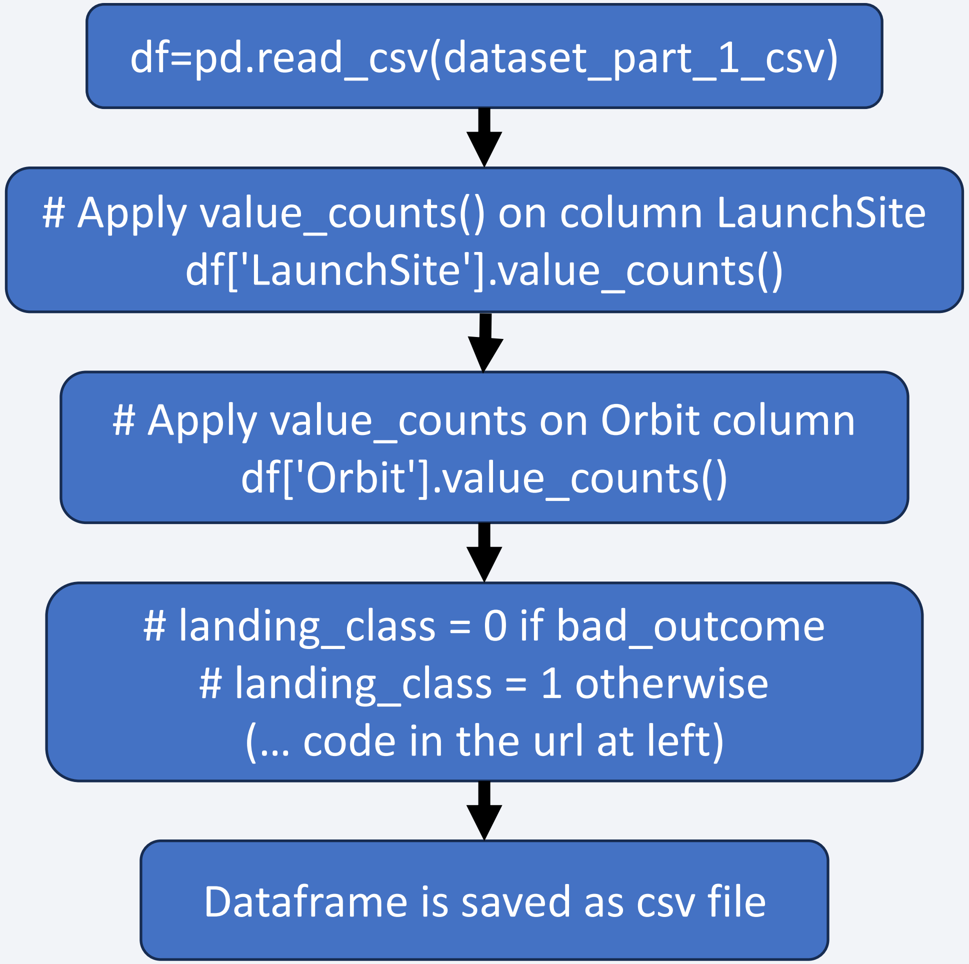
Data Collection - Scraping

- A web page (Wikipedia) is accessed through the “requests” library –> Page contents is parsed by “BeautifulSoup”, which extracts a (HTML) “table” object -> this table is iterated and contents kept in a pandas ‘dataframe’ -> this dataframe is saved as csv.
- Full Jupyter Notebook is available here.



Data Wrangling

- Data (CSV format) saved in previous steps is loaded for a pandas dataframe -> Some summary calculations are performed -> A new label (column) is created, to ease later analysis -> Updated table is saved as CSV file
- Full Jupyter Notebook is available here.



EDA with SQL

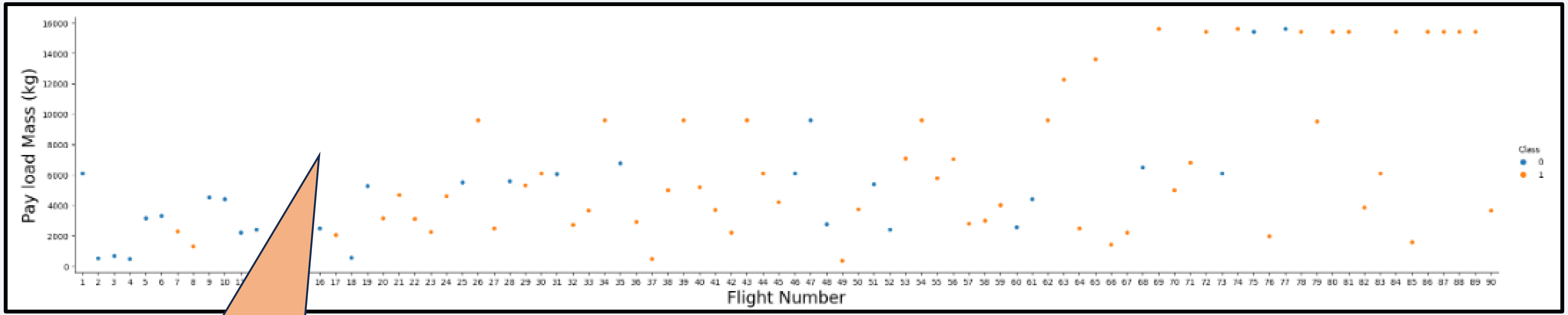
- Connection to the data base (through SQLAlchemy)
- Filtering records with date null
- Findind unique launching locations
- Finding total payload for customer NASA (CRS)
- (..)
- **Ranking the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.**

Queries executed:

- %sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
- %sql select distinct Launch_Site from SPACEXTABLE;
- %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer='NASA (CRS)'
- %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_version='F9 v1.1'
- (…)
- %sql SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' and (Landing_Outcome = 'Failure (drone ship)' or Landing_Outcome = 'Success (ground pad)') GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC

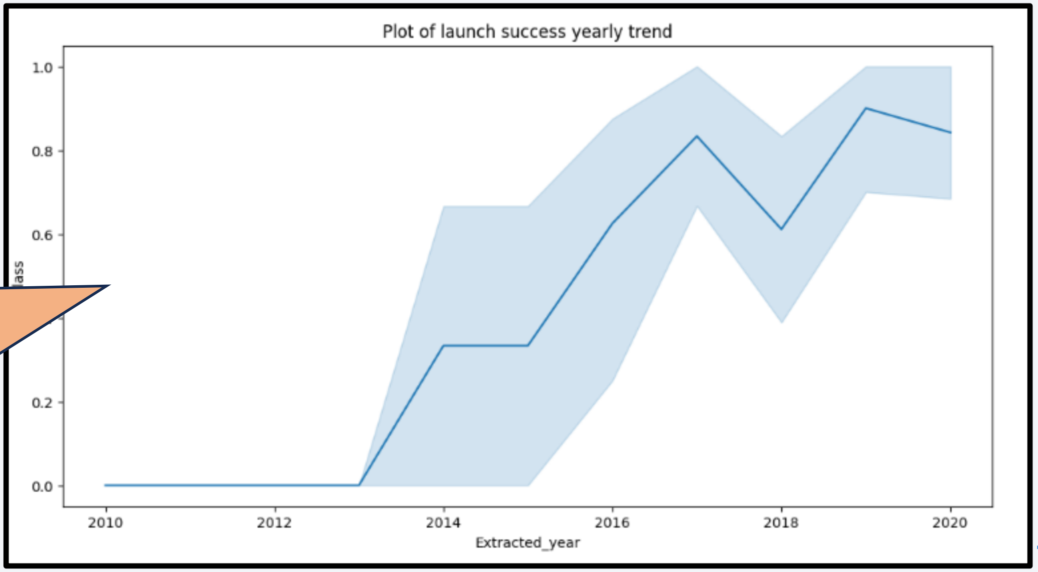
Full Jupyter Notebook is available here.

EDA with Data Visualization



How the ability to transport heavier payloads has evolved over time

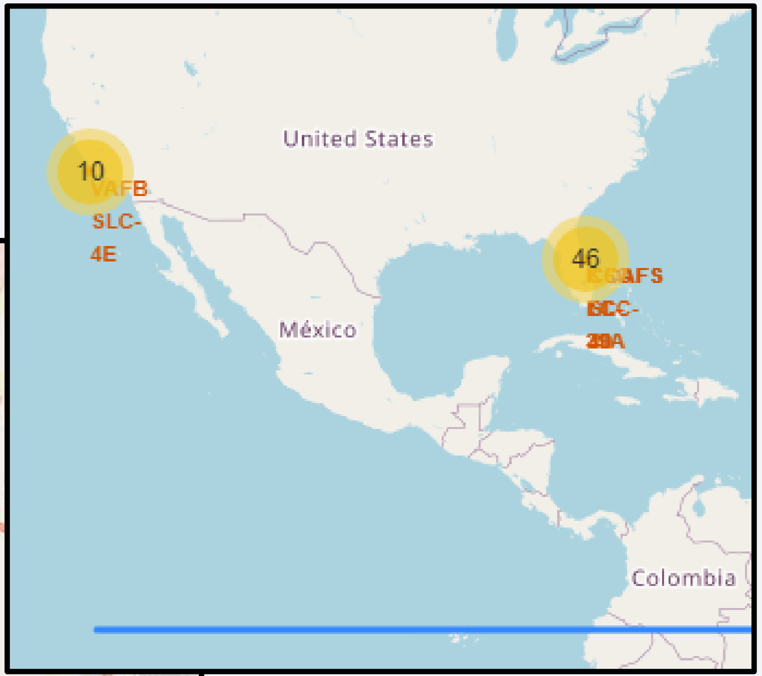
How success rate of 1° stage rocket recovery has evolver over time



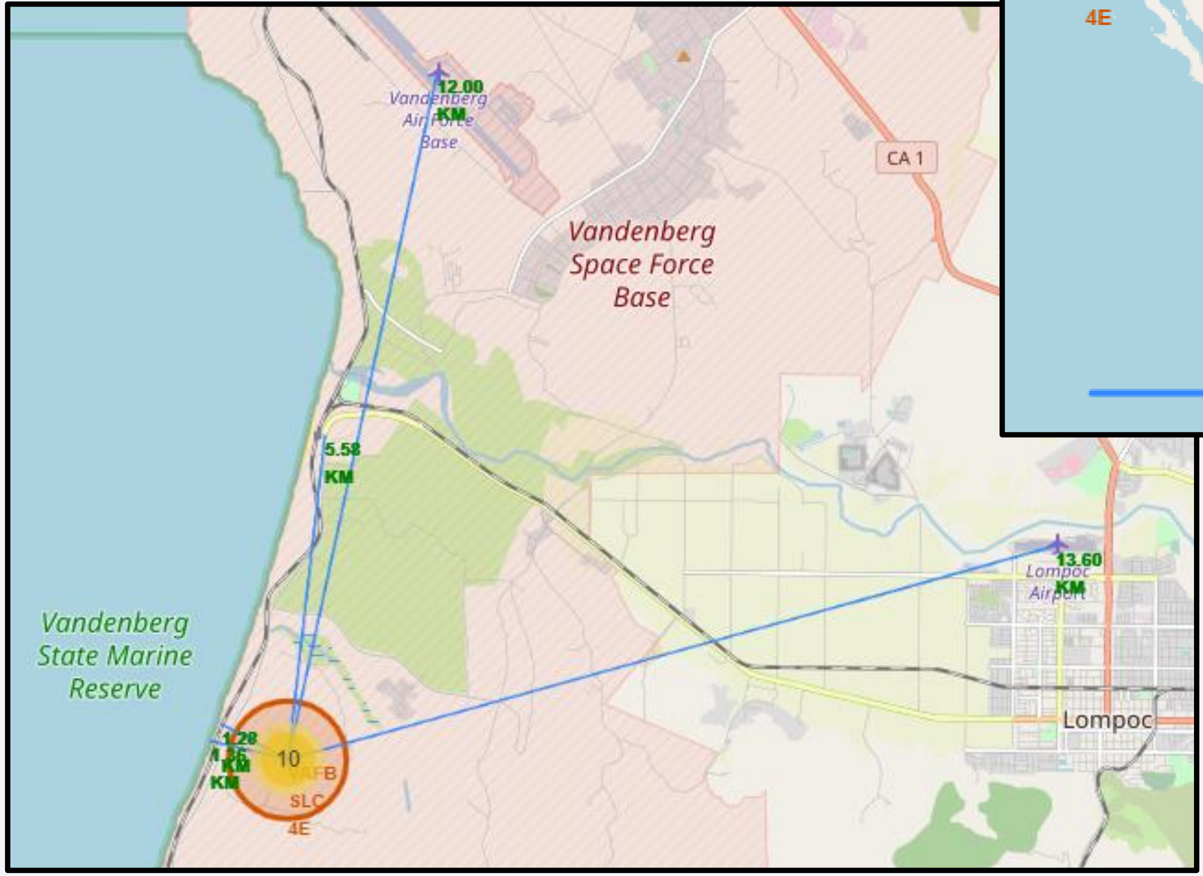
Full Jupyter Notebook is available here.

Build an Interactive Map with Folium

Launch locations, number of launches and Equator ☞ (blue line) (zoom allows to access to further details)



Distance of the launch location to costal line (1,36km), railway (1.3km), highway (5.6km) and two nearest airports (12.0 km and 13.6 km) ☞

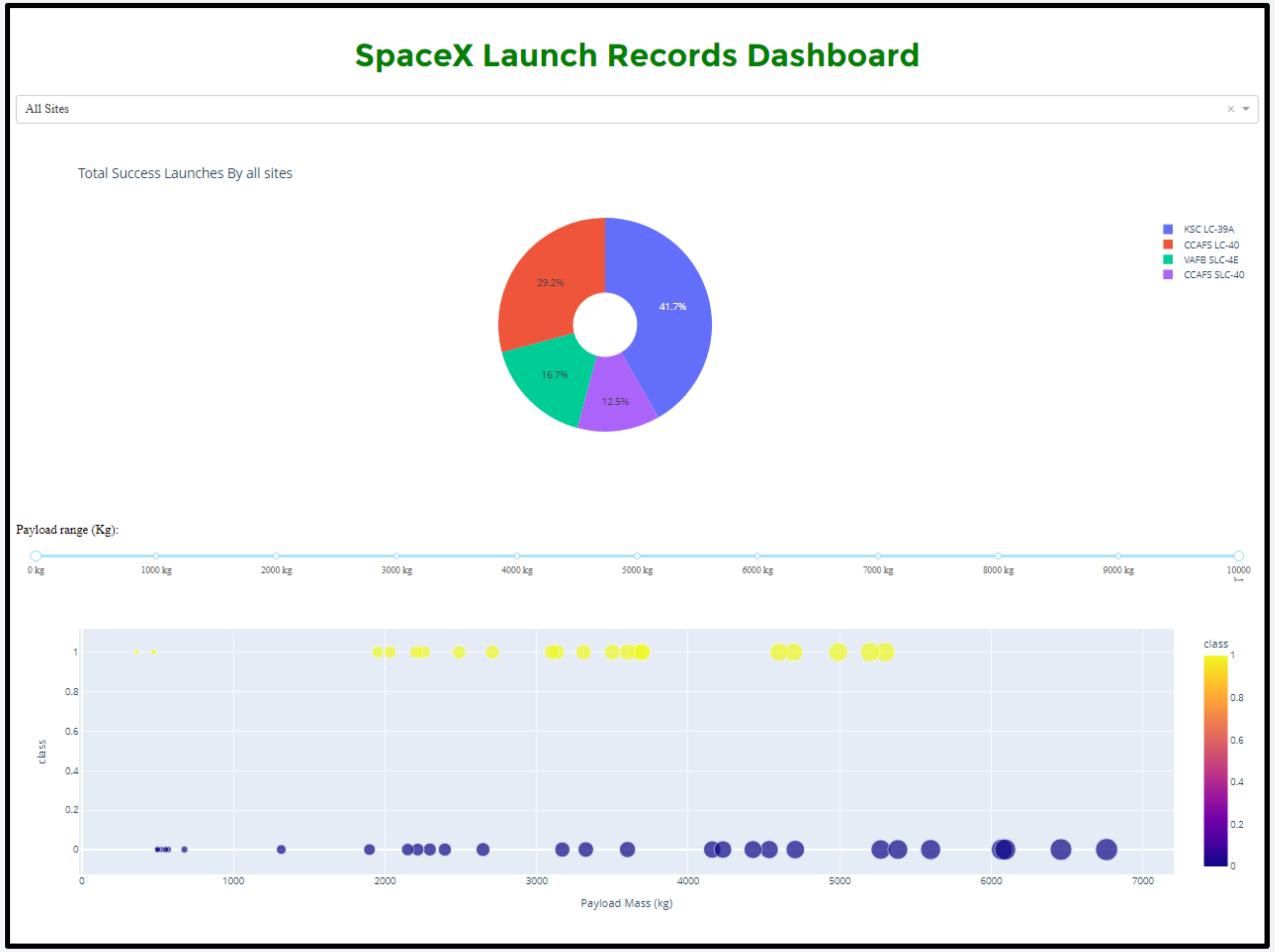


Full Jupyter Notebook is available [here](#).

Build a Dashboard with Plotly Dash

- Total Success grouped by ☞ launching site (top) was added to allow to assess the relevance of the launching site in the recovery of the 1st stage of the rocket – users can select all sites or just one of them

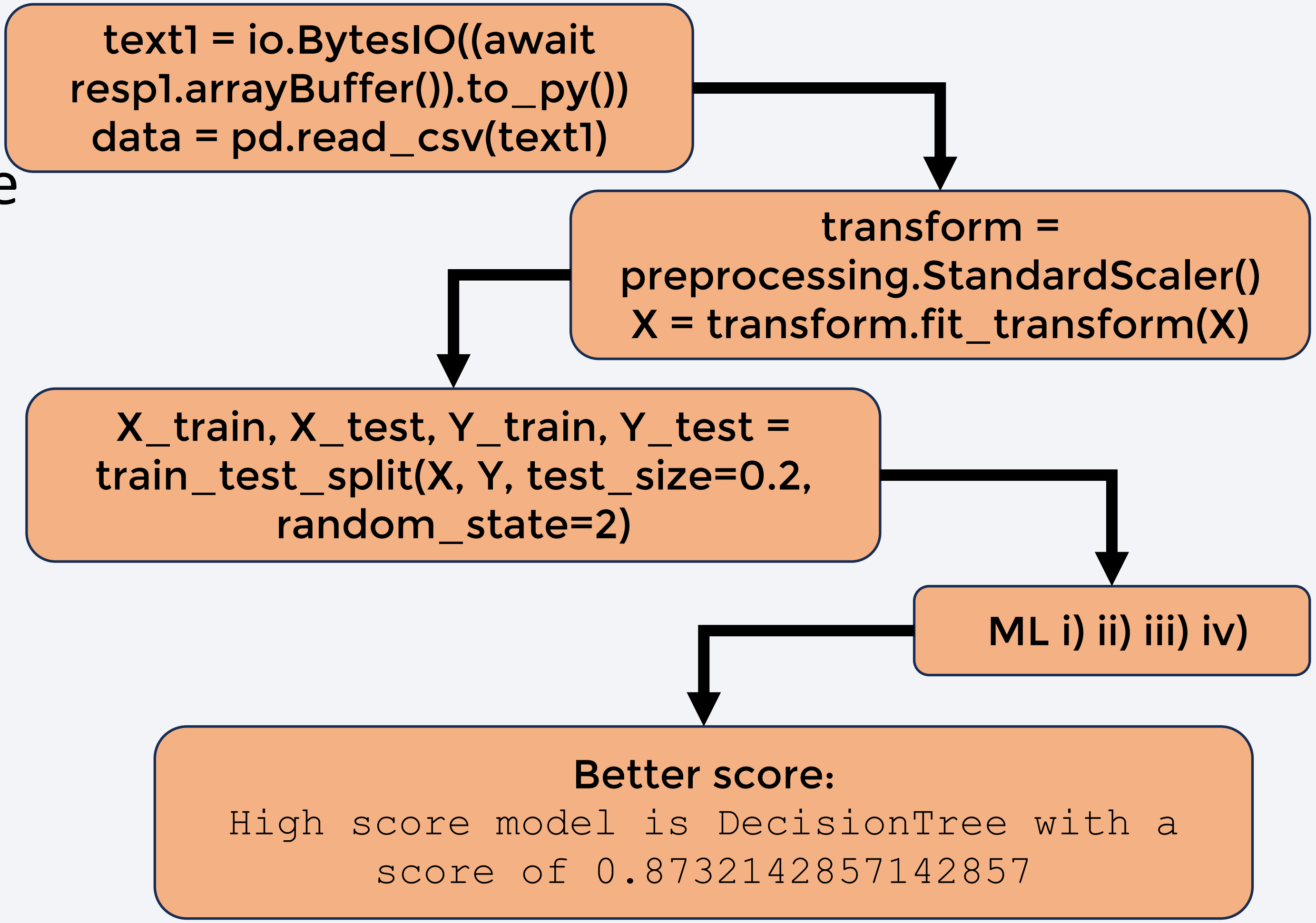
- Success by Payload mass (kg) ☞ (bottom) allow to assess the relevance of the payload – users can adjust (slider bar) the range of analysis



Full Python code is available [here](). Data file (csv) [here]().

Predictive Analysis (Classification)

- Data read from csv file and loaded into a Pandas dataframe
- Data standardization
- Train / test spli
- Application of i) Logistic Regression; ii) Support Vector Machine; iii) Decision Tree; iv) K-Nearest Neighbors -> Score + confusion matrix
- Check for the model with best score

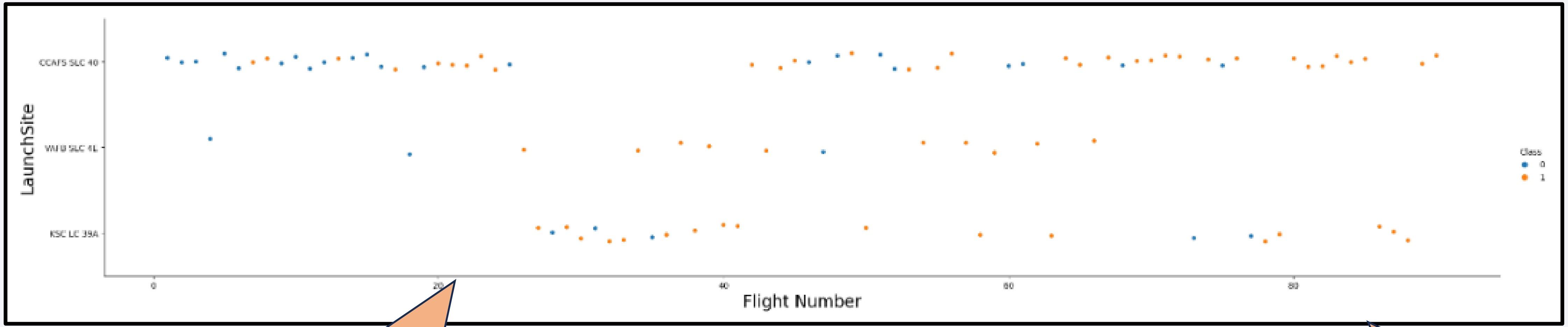


Full Python code is available here.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

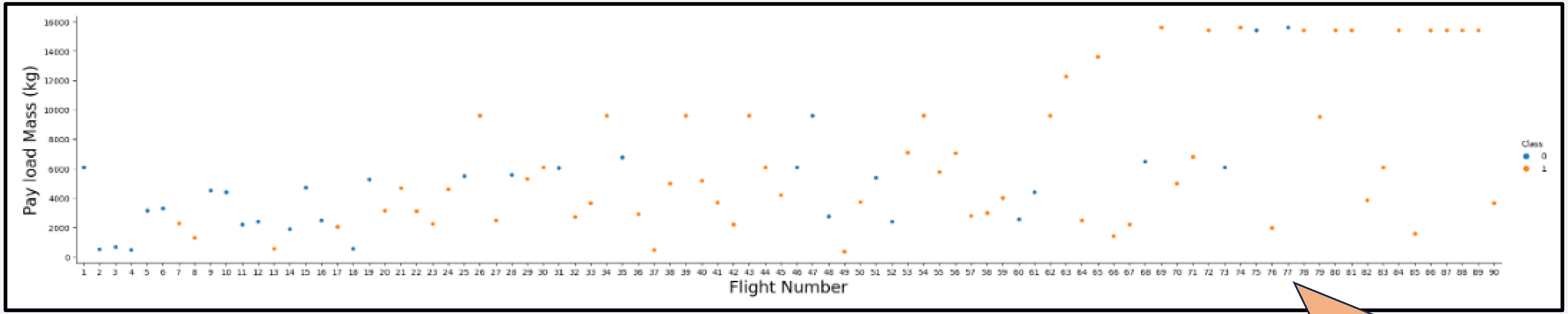


CCAFS LC-40 was mainly used for the first series of flights, ans it is still the most used Launching site

Field expert insights are needed to interpret this trend

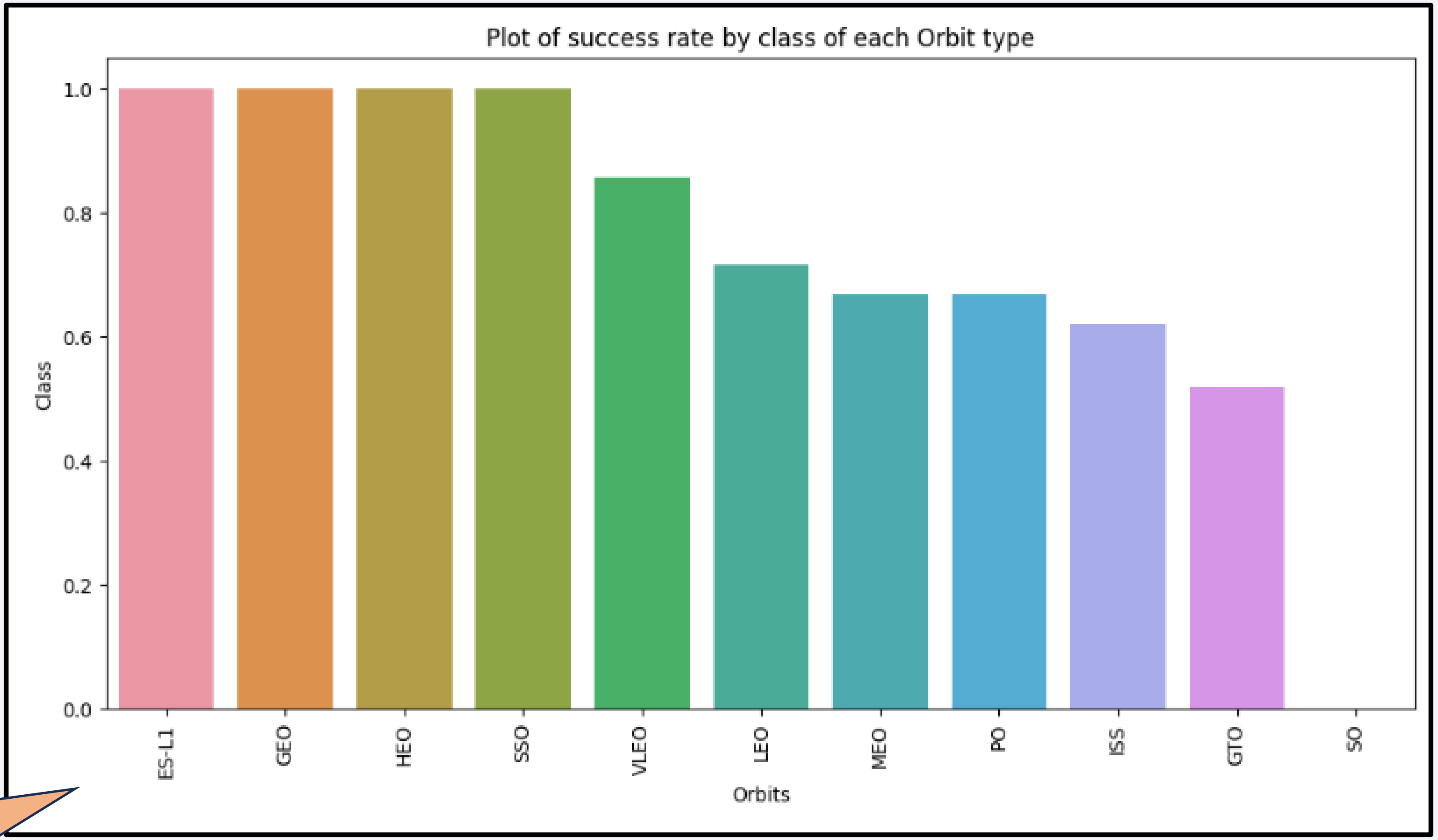
First flights had a much lower success rate than later ones

Payload vs. Launch Site



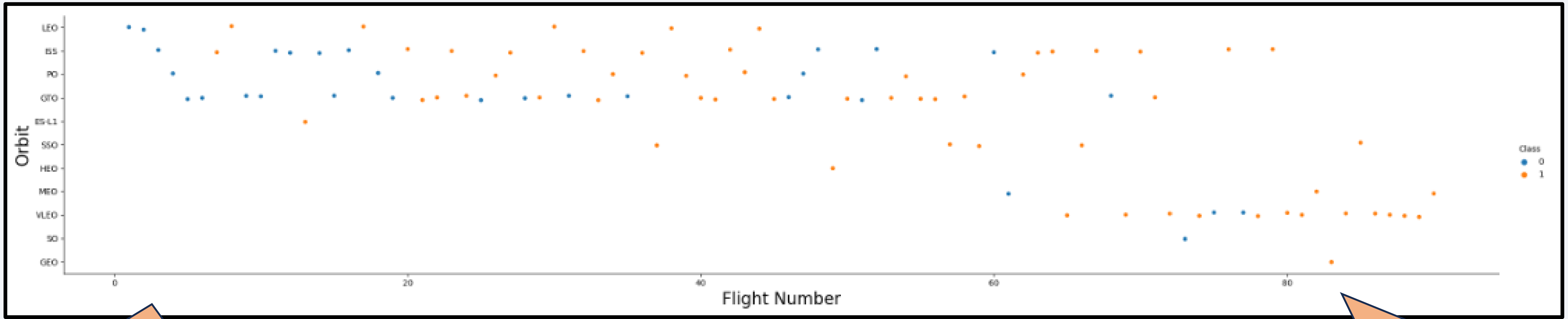
Payload – and success rate – increased with flight number / time

Success Rate vs. Orbit Type



ES-L1, GEO, HEO and SSO orbits have the most successful landing score

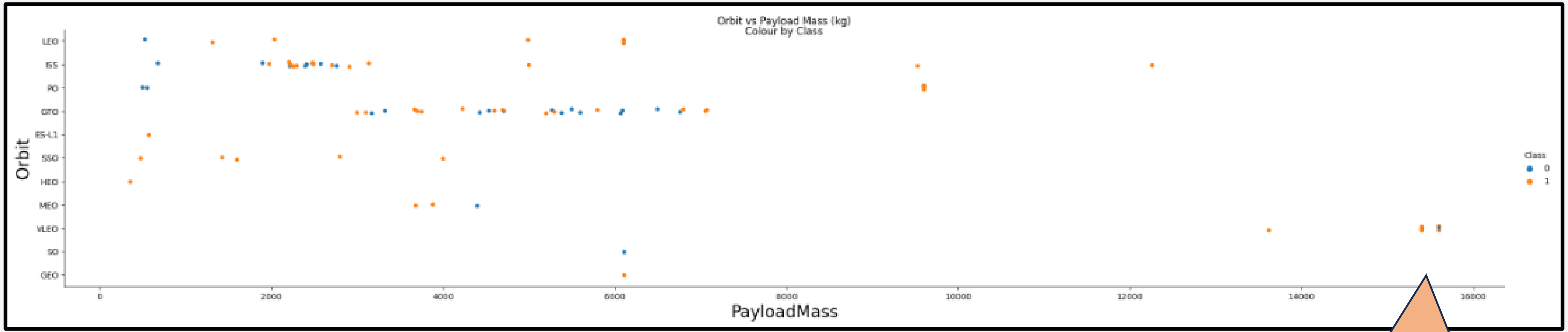
Flight Number vs. Orbit Type



First '40 flights' have targeted LEO, BS, PO and GEO orbits

GEO, SO, VLEO and HEO orbits have been tried only in the 'last' flights

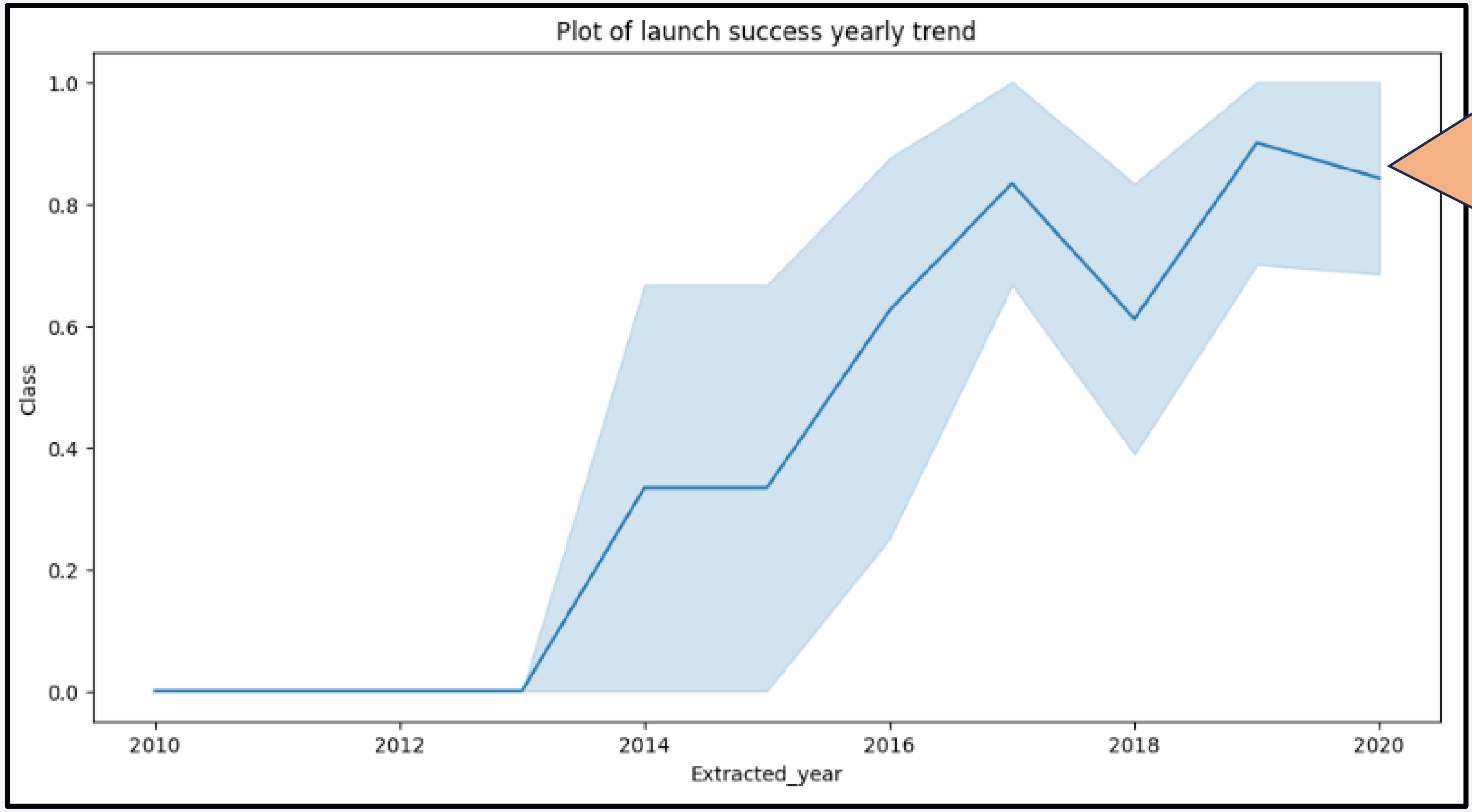
Payload vs. Orbit Type



Bigger payloads were for VLEO (Stralink system?)

Field expert insights are needed to interpret this trend

Launch Success Yearly Trend



Success rate increased «linearly» from 2013 to 2017, and slower later.

Fine tuning and bigger efforts will probably need to maintain or increase this success rate.

All Launch Site Names

The clausule ‘distinct’ filters the select results and shows only distinct values.

RESULTS

Task 1

Display the names of the unique launch sites in the space mission

```
#%sql select * from SPACEXTABLE limit 2;
%sql select distinct Launch_Site from SPACEXTABLE;
```

```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

The clausule `like 'CCA%'` filters only the 'Launch_Site' starting by «CCA», and the clausule '`limit 5`' lists only the first 5 ocurrences.

RESULTS

Display 5 records where launch sites begin with the string 'CCA'

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

```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO
2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)

Total Payload Mass

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

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer='NASA (CRS)'

 * sqlite:///my_data1.
Done.
```

sum(PAYLOAD_MASS__KG_)
45596

`Where Customer='NASA (CRS)'` **selects only the records of this** `Customer`

`SUM(Payload)` **computes the sum of** «`PAYLOAD_MASS__KG_`» **values for the selected records**

RESULTS

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_version='F9 v1.1'

 * sqlite:///my_data db
Done.
```

avg(PAYLOAD_MASS__KG_)
2928.4

`Where Booster_version='F9 v1.1'` filters only the records for this `Booster_version`

`avg(Payload)` computes the average of the «`PAYLOAD_MASS__KG_`» values for the selected records

RESULTS

First Successful Ground Landing Date

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
#%sql select * from  SPACEXTABLE limit 20;
%sql select min(Date) FROM SPACEXTABLE WHERE Landing_Outcome like "Success (ground pad%";
```

```
 * sqlite:///my_data   db
Done.
```

min(Date)
2015-12-22

`WHERE Landing_Outcome like "Success (ground pad%"` **filters only the records where** `Landing_Outcome` **starts width «**`Success (ground pad`**»**

`min(Date)` **finds the minimum / first** `Date` **for the selected records**

RESULTS

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql SELECT Booster_Version\
   FROM SPACEXTABLE\
   WHERE Landing_Outcome = 'Success (drone ship)' \
   AND Payload_Mass__KG_ > 4000\
   AND Payload_Mass__KG_ < 6000
```

```
 * sqlite:///my_data1.db
Done.
```

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

- List of conditions to be satisfied
- Multiline SQL command (lines continuated by "\" at the end of the line

RESULTS

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql select * from ( \
        (select count(*) as Success from SPACEXTABLE where Mission_Outcome like '%Succ%') as T1,\
        (select count(*) as Failure from SPACEXTABLE where Mission_outcome not like '%Succ%') as T2\
            )
```

```
 * sqlite:///my_data1.db
Done.
```

Success	Failure
100	1

- Outer Select from
- Two inner Selects

RESULTS

Boosters Carried Maximum Payload

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT Booster_Version, Payload_Mass__KG_ \
    FROM SPACEXTABLE\
       WHERE Payload_Mass__KG_ = (\
            SELECT MAX(Payload_Mass__KG_) FROM SPACEXTABLE\
                                   )\
ORDER BY Booster_Version
```

* sqlite:///my_data1.db

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

- 1 outer Select from
- 1 inner Selects

RESULTS

2015 Launch Records

```
%sql select strftime('%m', Date) as Month, Landing_Outcome, Booster_Version, Launch_Site, Date\
     from SPACEXTABLE\
         where Date like '%2015%'\
         and Landing_Outcome like '%Failure%'
```

 * sqlite:///my_data1.db
Done.

Month	Landing_Outcome	Booster_Version	Launch_Site	Date
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	2015-10-01
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	2015-04-14

- Format Date as month number
- Filtering Failure for 2015

RESULTS

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

```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome)\
      FROM SPACEXTABLE\
         WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'\
         and ( Landing_Outcome = 'Failure (drone ship)' or Landing_Outcome = 'Success (ground pad)')\
         GROUP BY Landing_Outcome\
     ORDER BY COUNT(Landing_Outcome) DESC
```

```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	COUNT(Landing_Outcome)
Success (ground pad)	5
Failure (drone ship)	5

- **Filtering with complex conditions (`and` and `or`)**
- **Sorting by `COUNT(Landing_Outcome)`**

RESULTS

Section 3

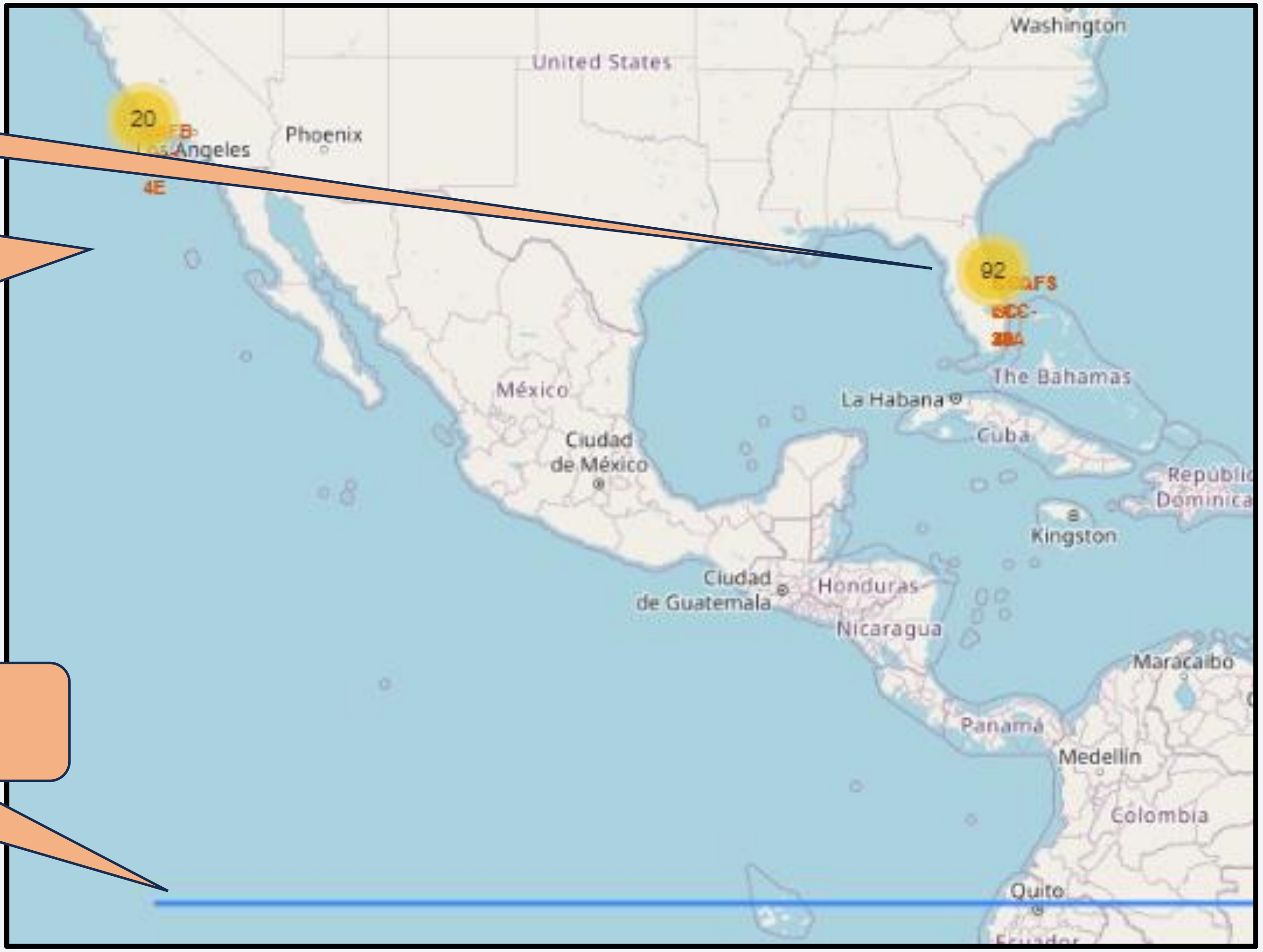
Launch Sites Proximities Analysis

Geographical Location of launch sites

Launch locations, number of launches:

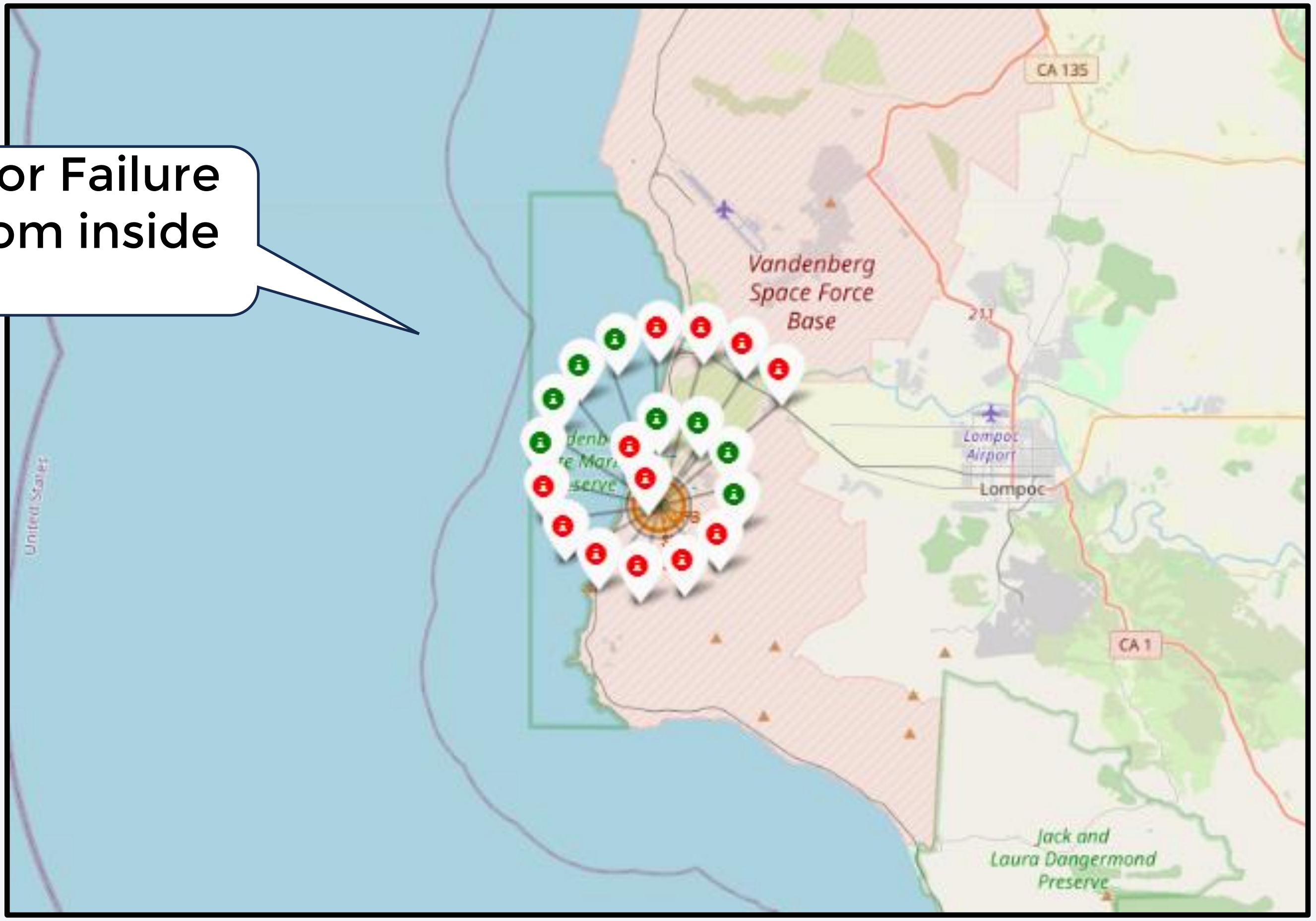
- All launchsites located near the coastline

Equator (blue line)



Markers on launch sites colored by success

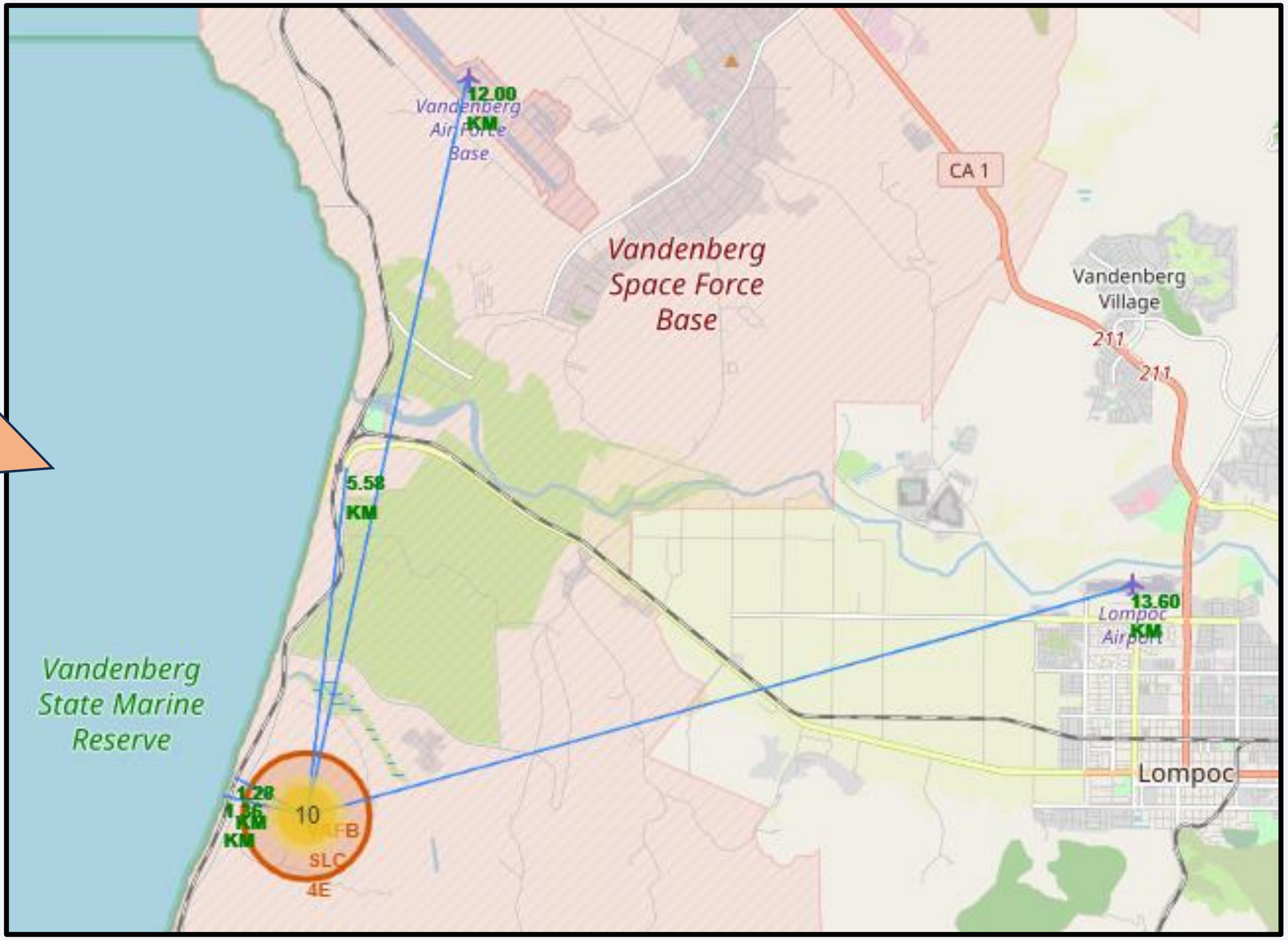
- Green used for Success, red for Failure
- Launches in spiral growing from inside ordered by Date



Distances from Launch site to landmarks

Distance of the launch location to costal line (1,36km), railway (1.3km), highway (5.6km) and two nearest airports (12.0 km and 13.6 km)

- Launch site located near main transportation facilities



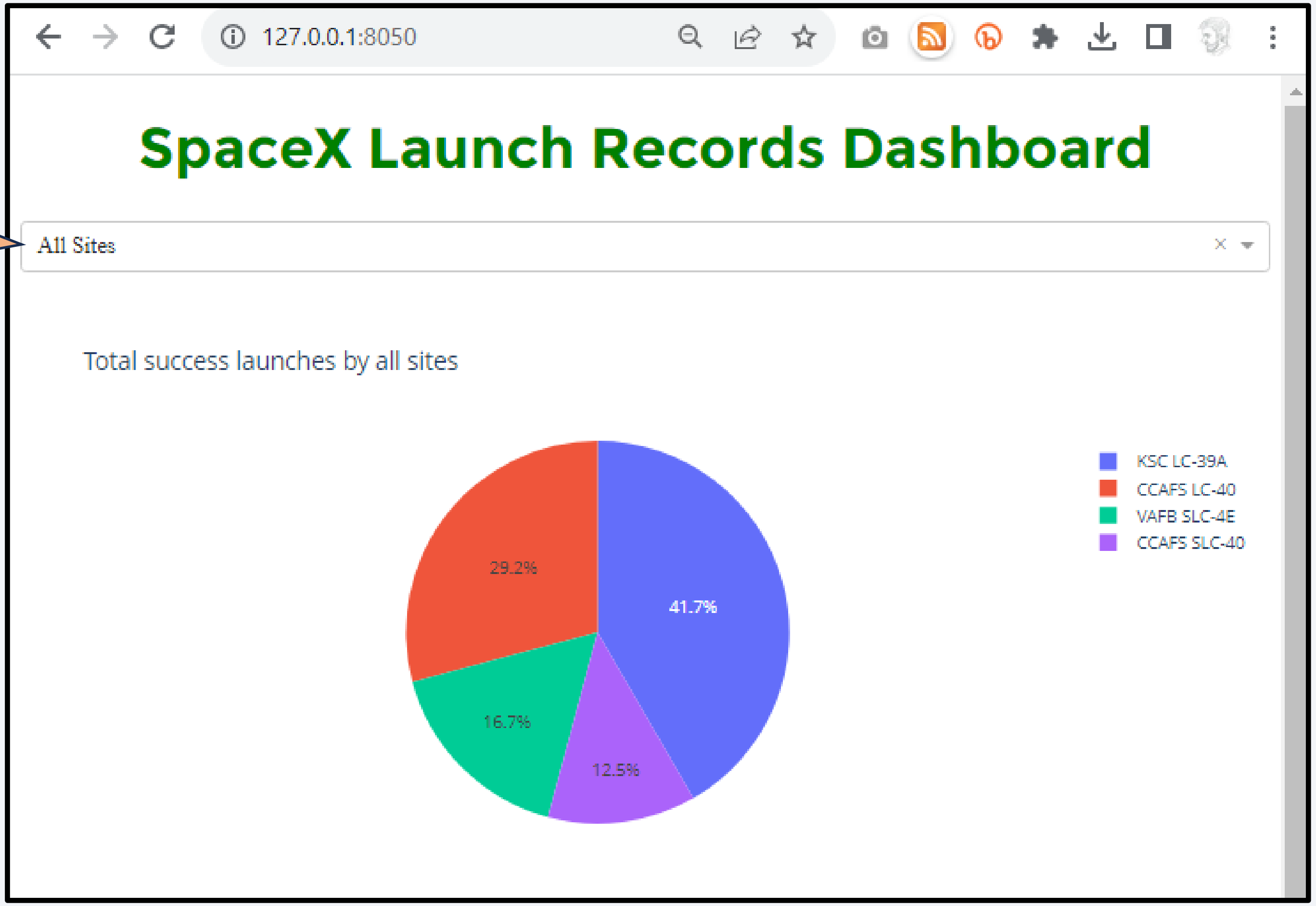
Section 4

Build a Dashboard with Plotly Dash

Interactive Dash board

Launch Success for all Launching sites

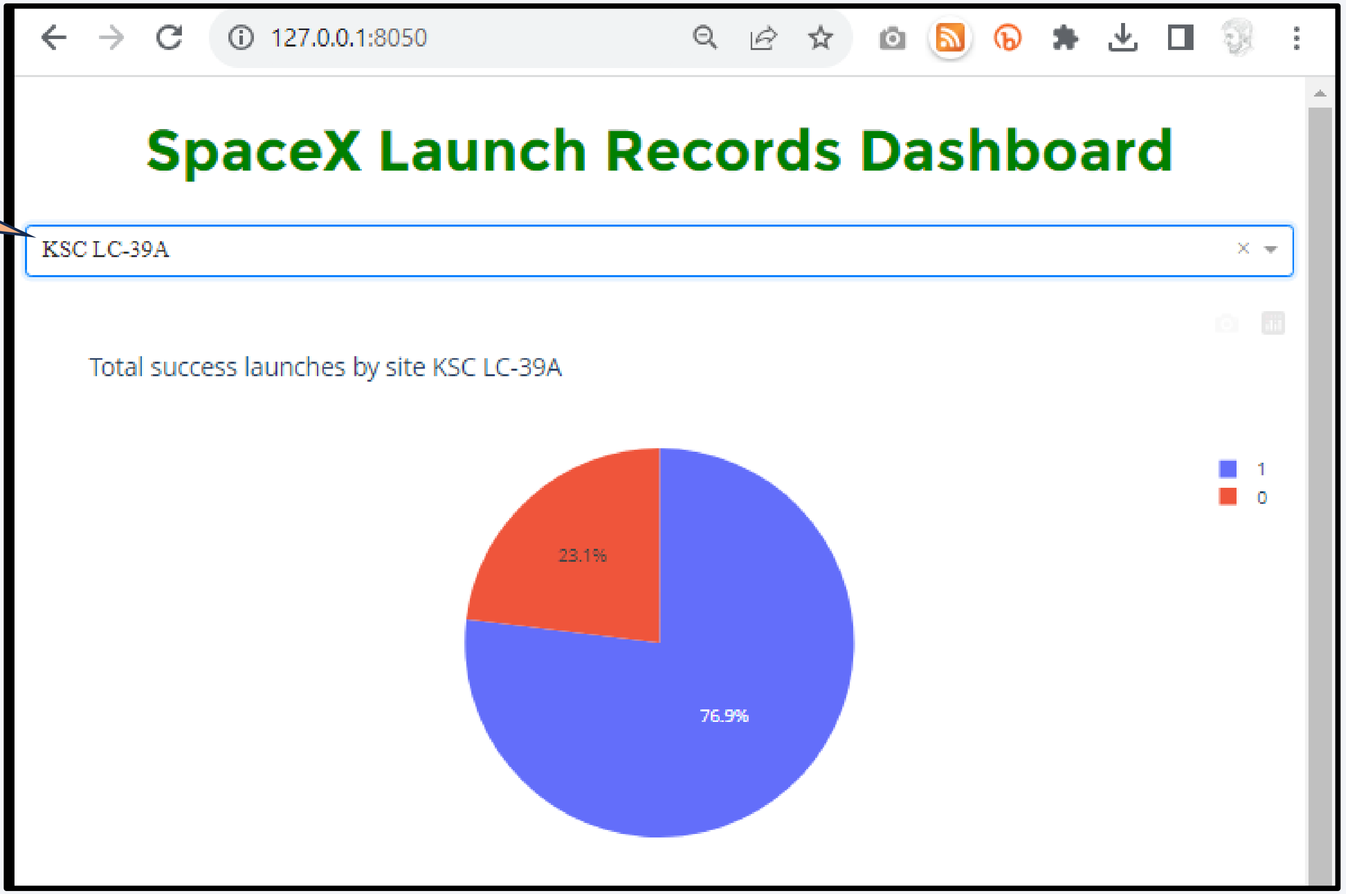
- Most successful site is KSC LC-39A



Launch site with highest launch success

KSC-LC 39 A
(more info here)

- Success rate of 70%



Success vs. payload colored by Launch Site
(bubble size by payload)

Low-range (0-5000 kg) of Payload Mass

Field expert insights are needed to improve this Dash board

High-range (5000-10000 kg) of Payload Mass

Section 5

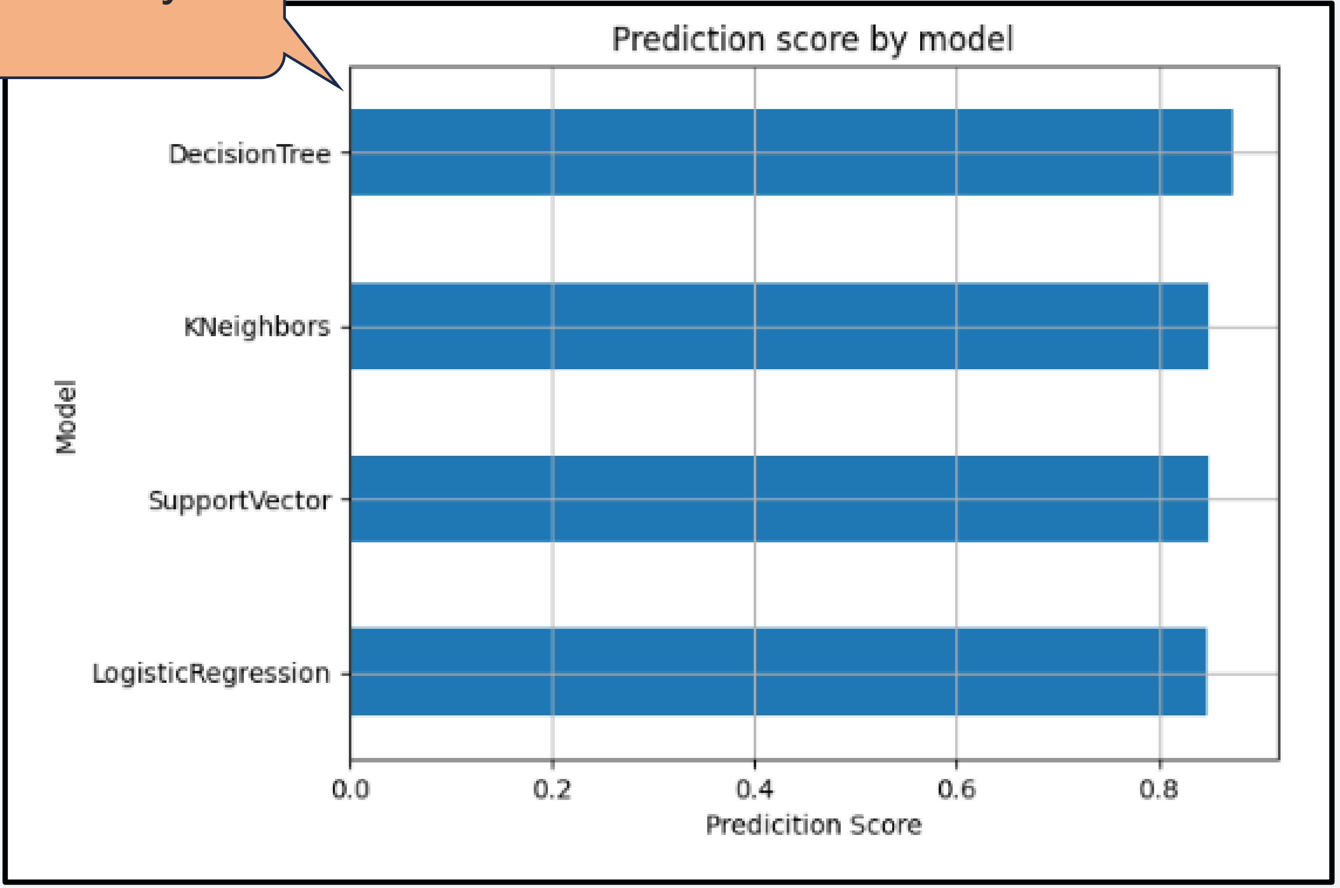
Predictive Analysis (Classification)

Classification Accuracy

Barplot sorted by Score

	Score
LogisticRegression	0.846429
DecisionTree	0.873214
SupportVector	0.848214
KNeighbors	0.848214

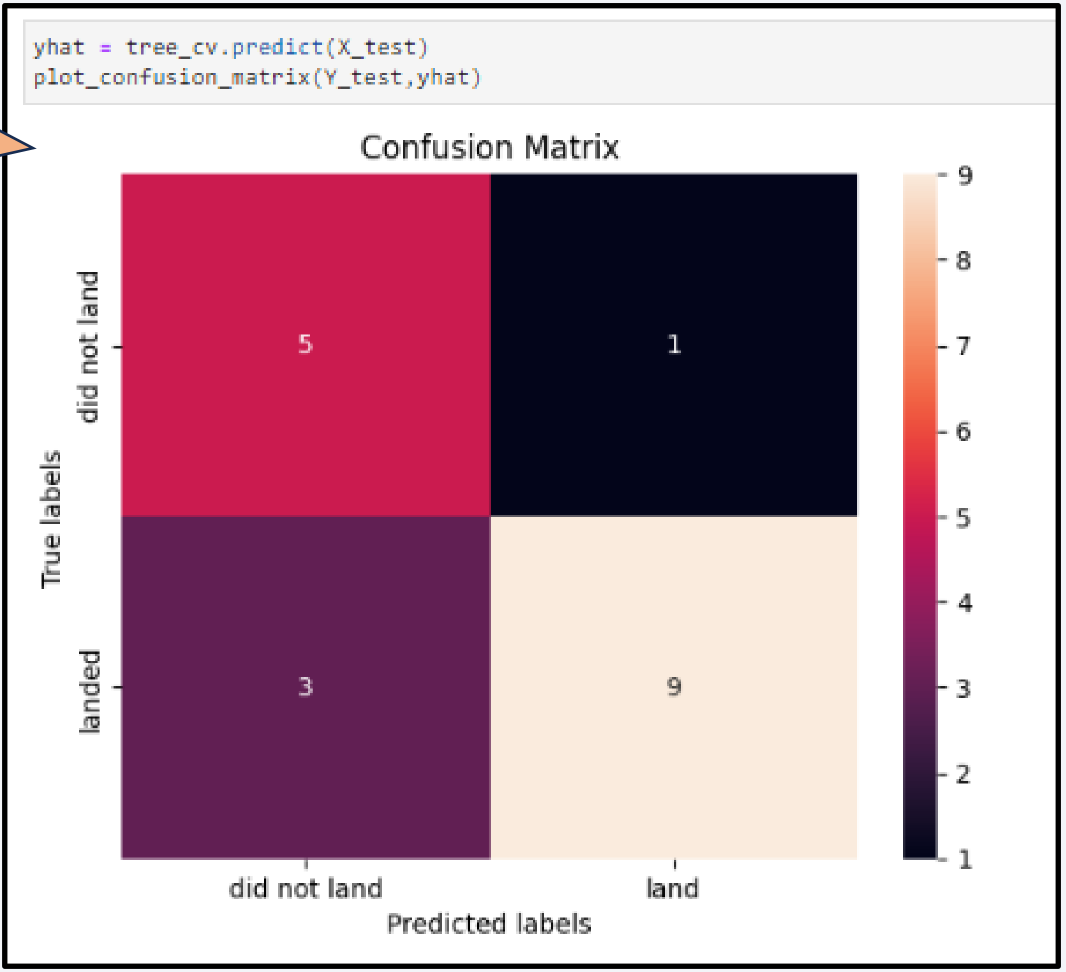
ML Score for different methods



Confusion Matrix

14 predictions right, 4 wrong.

Main issue with false negatives – 3 predictions of 'not landing' were wrong – the rocket did land successfully.



Conclusions - I

- Data regarding Falcon 9 rocket launchingrevealed some features
  - A trend on success rate increase over time (apparently with a plateau after 2018~2019)
  - A trend on the type of orbits used and the increase of payload mass over time

  Some regularities about the launching locations:
    - Near the coastline
    - and major landmarks as railways, airports and highways (ease of transportation)
  - And some correlations as
    - Number of launches (for the same site) and success rate
    - Best ML algorithm for prediction is “Decision Tree”

Field expert insights (**weather conditions?**) are needed to improve this analysis.

Conclusions - II

- We have been able to present an example of Data Science application, from data harvesting to prediction (through machine learning) based on the available data
- This capstone project is very well aligned with the contents of the "IBM Data Science» course specialization at Coursera – Thank you very much and Congratulations!

Appendix

- Some details on Jupyter notebook programming

```
response = requests.get(static_json_url)
response.status_code
#print(response.content)
response.content[0:100]
```

```
b'[{"fairings": {"reused": false, "recovery_attempt": false, "recovered":
```

Echoing just a small part of a text var 'just to check'

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] == 'Falcon 9']
data_falcon9
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None

Instead of filtering != "Falcon 1", I decided to filter by "Falcon 9"

Appendix

- Some details on Folium

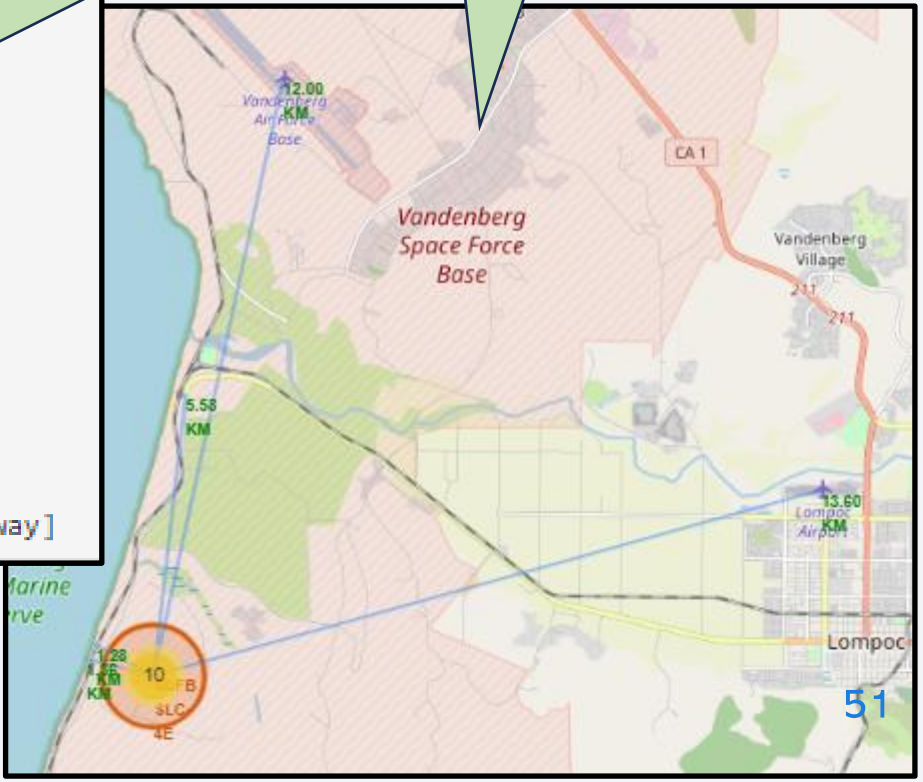
Defined a scalable approach to draw several destinations from a central point

```
c_Lompoc_airport=[34.66562, -120.4675028]
c_Vanderberg_airport=[34.73821, -120.58272]
c_coastline=[34.63582, -120.62508]
c_railway=[34.63819, -120.62306]
c_highway=[34.68269, -120.6038]

c_launchsite=[34.63284, -120.61072]


to_Lompoc_airport=[c_Lompoc_airport, c_launchsite]
to_Vanderberg_airport=[c_Vanderberg_airport, c_launchsite]
to_coast=[c_coastline, c_launchsite]
to_railway=[c_railway, c_launchsite]
to_highway=[c_highway, c_launchsite]

to_dest=[to_Lompoc_airport, to_Vanderberg_airport, to_coast, to_railway, to_highway]
```



Appendix

- Data framing and matlibplot

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

print(models)
```

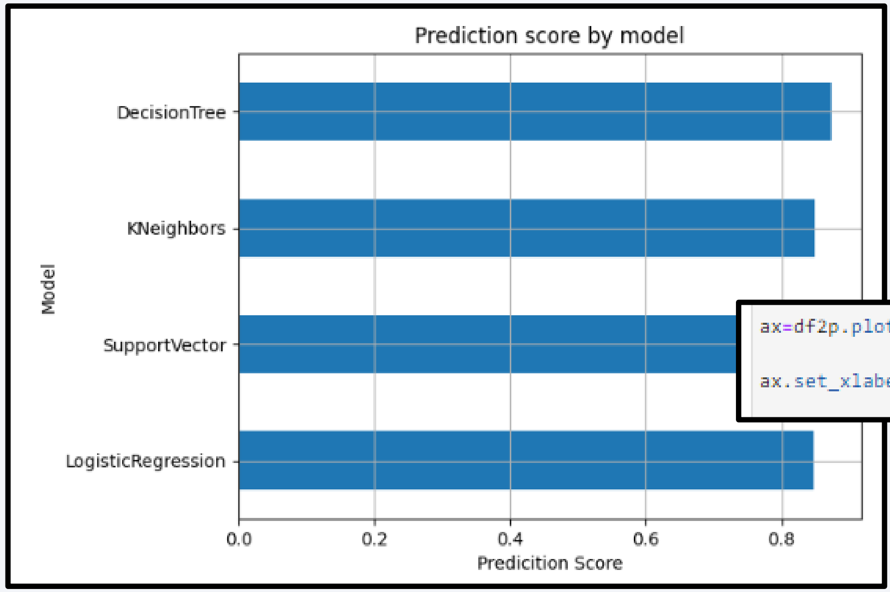
From 'models' to df2p

```
am=[]
av=[]
for i in models:
    print(i)
    am.append(i)
    av.append(models[i])
print(am, av)

df2=pd.DataFrame({"Model":am,
df2p=df2.sort_values('Score')
df2p
```

```
LogisticRegression
DecisionTree
SupportVector
KNeighbors
['LogisticRegression', 'Decisi
2858]
```

	Model	Score
0	LogisticRegression	0.846429
2	SupportVector	0.848214
3	KNeighbors	0.848214
1	DecisionTree	0.873214




```
ax=df2p.plot(x="Model", y="Score", kind="barh",\
             title='Prediction score by model', sort_columns=True, grid=True, legend=False)
ax.set_xlabel("Prediction Score")
```

... To barchart ordered by Score

Acknowledgments

I wish to express my appreciation to

- the quality and performance of Coursera platform and overall service
- the quality of the contents and learning activities designed by IBM
- the possibility to freely try online tools as Cognos and Watson Studio (IBM Cloud)

One very special word goes for all my online colleagues, both reviewers and reviewees – neverthless the quality of the contents, MOOC learning is always improved by human interaction – please feel free to contact me at https://www.linkedin.com/in/pedropimenta/

Thank you!