IBM Developer
SKILLS NETWORK

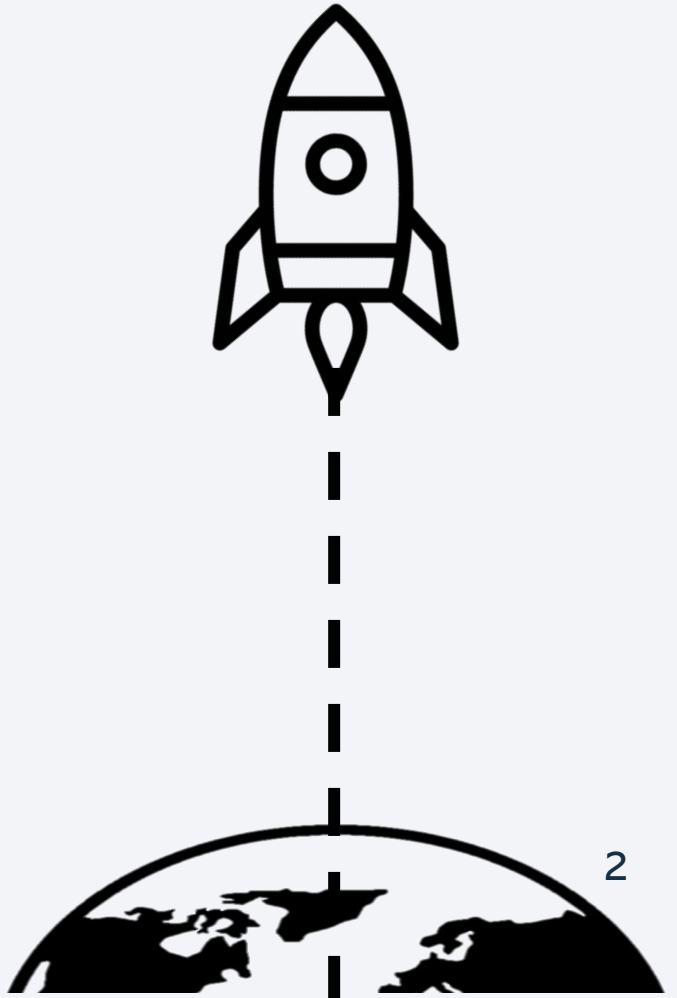
Winning Space Race With Data Science

Eva Villar Álvarez
Data Scientist

08/03/2024

https://github.com/EvaTartaruga/SpaceRacewithDataScience.git

Outline



Executive Summary

The Problem

SpaceX is making rocket launches relatively inexpensive, unlike other rocket providers.

How?

SpaceX's Falcon 9 can recover the first stage

The Question:

Can the first stage of the SpaceX Falcon 9 rocket will land successfully?

Methodologies to solve the question

Data collection and cleaning

Request to the SpaceX API

Web scraping related Wiki pages

Preprocessing data (cleaning data)

Exploratory Analysis

Data wrangling & identify variables

EDA with SQL

EDA with data visualization

Interactive map with Folium

Dashboard with Plotly Dash

Data modeling

Predictive analysis (Classification)

- Logistic regression
- SVM
- KNN
- Regression Tree

Summary of Results

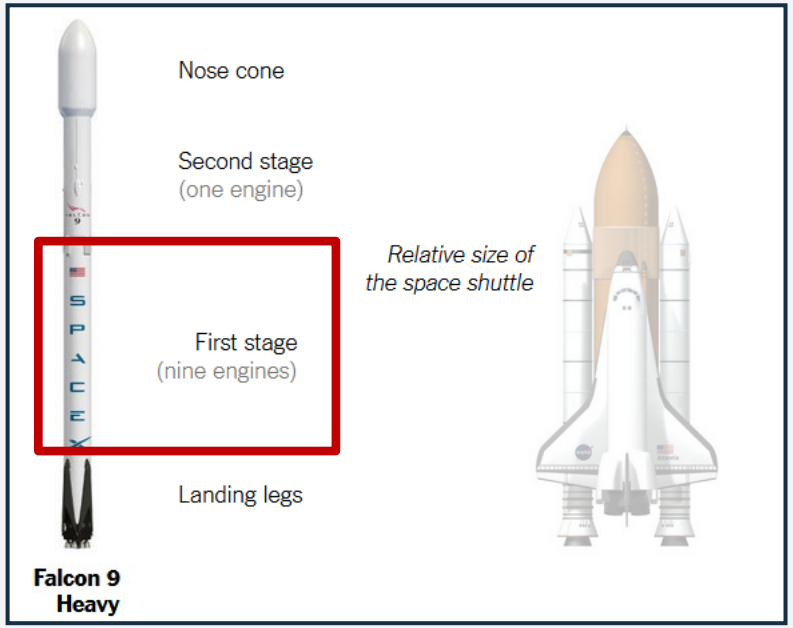
Insights drawn from EDA	• Analysis of outcome correlation with several variables • Scatter, Bar graphs, and Line plots
SQL Queries	
Geospatial Analysis	
Dashboard interactive EDA	
Machine Learning Classifiers (Predictive analysis)	• Classification accuracy • Confusion matrix • ROC and Precision-Recall curves

Introduction

Context

- Falcon 9 SpaceX's rocket launches with a cost of 62 million $
- Other providers cost upward of 165 million $ each.
- Savings is because SpaceX can reuse the first stage.

- SpaceY company wants to bind against SpaceX for a rocket launch.
  Therefore, if we can determine if the first stage will land,
  we can determine the cost of a launch

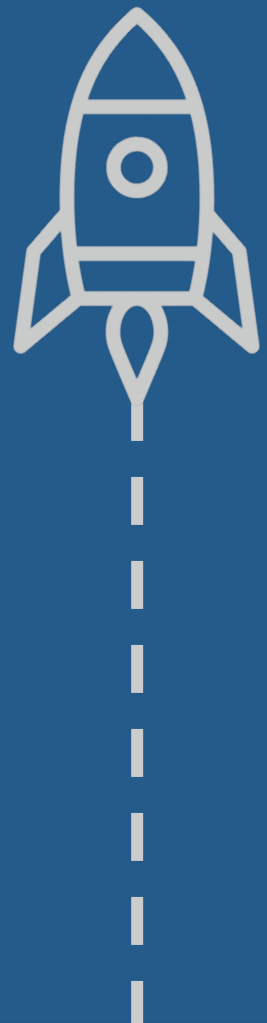


Problems to answer if the rocket will land successfully

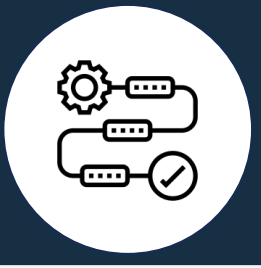
- What variables will have more impact in determining the success rate of a successful landing?
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate?

Section 1: Methodology

- Data Collection
- Data Wrangling
- Data Exploratory Analysis
- Data Prediction



Methodology



Data collected from:

- SpaceX Rest API
- Web scraping related Wiki pages

Data wrangling

- Identify missing values, and categorical and numerical variables
- Identify Target variable vs predictor variable
  - Explore the occurrence of predictor variables
  - Explore and prepare the target variable

Perform exploratory data analysis (EDA) using visualization and SQL

- Graph target and attributes to identify patterns and trends with Matplotlib
- Gather information with SQL queries

Perform interactive visual analytics using Folium and Plotly Dash

- Geographical analysis for launch sites, outcome and proximity to landmarks
- Interactive Analysis to quickly explore data using visualizations

Perform predictive analysis using classification models

- Featuring Engineering:
  - Data normalization,
  - Hyperparameter Tuning (gris-search CV)
  - Cross Validation,
  - Multiple Algorithms models
- Evaluation:
  - Test accuracy score,
  - confusion matrix,
  - ROC and Precision-Recall curves

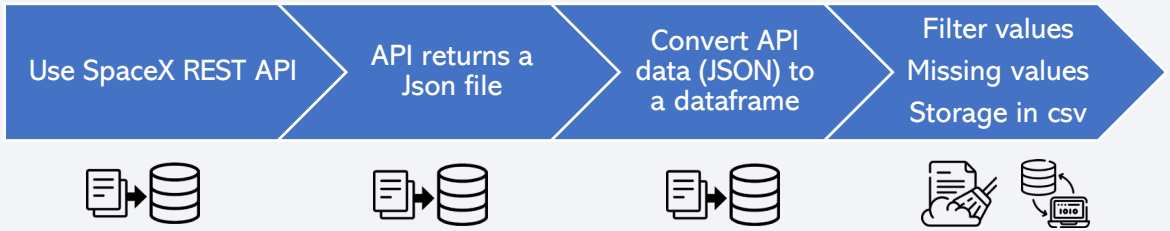
Data Collection

SpaceX Rest API

- **Source:** https://api.spacexdata.com/v4/launches/past
- **Method:** using the GET request
- **Information:** rocket, payloads, cores, launch specifications, landing specifications, and landing outcome

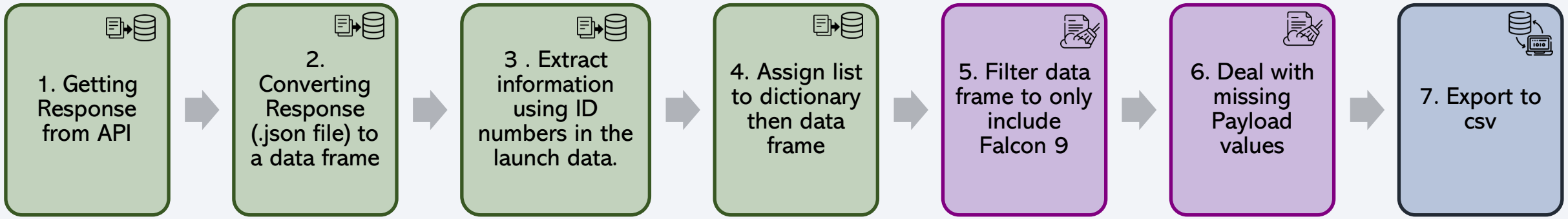
Web Scrapping

- **Source:** https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- **Method:** Python BeautifulSoup package
- **Information:** Falcon 9 historical launch records



Data Collection – SpaceX API

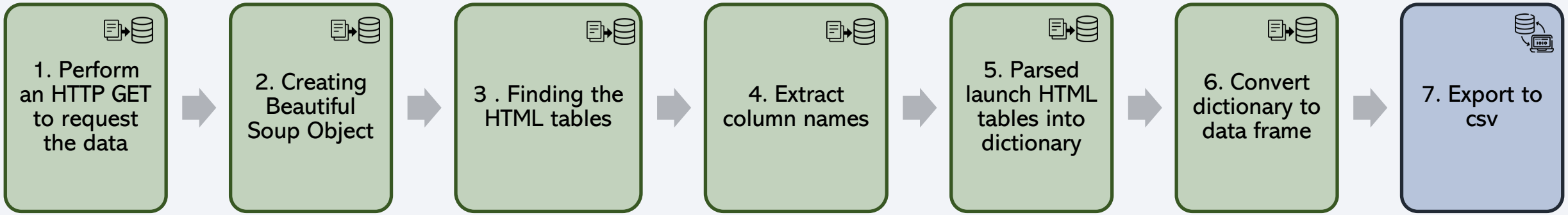
*See Appendix A for code



https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/A-Spacex-data-collection-api.ipynb

Data Collection - Scraping

*See Appendix B for code



https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/B-Webscraping.ipynb

Data Wrangling

Explore data format

- **Identify** missing values
- **Identify** categorical and numerical variables

Identify Target variable and predictor by reviewing attributes

- **Calculate** occurrence per type of attributes:
  - launch sites
  - orbit
- **Transform** the target variable "*Landing outcome*" from categorical to numerical
  - Failure='0', if False or None in the string
  - Success='1', otherwise (if True in the string)
- **Organize** target variable in a new column called "Class"

Store the data

https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/C-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Scatter Graphs

Scatter plots show how much one variable (y) is affected by another (x), and if there are a correlation.

It is desirable to know if the *Flight Number* and *Payload Mass* variables are correlated to *Launch Site* and *Orbit Type.* Also if they affect the *Launch Outcome*

- Flight Number vs. Launch Site
- Payload Mass vs. Launch Site

- Flight Number vs Orbit Type
- Payload Mass vs. Orbit Type

Bar Graphs

Bar charts compare categorical data. Horizontal axis shows the specific categories and the vertical one represents the measured value.

The aim is to compare the *Success Rate* of the launches by *Orbit Type,* to analyze any pattern.

- Success Rate vs. Orbit Type

Line Graphs

Line plots show trends between variables. They are commonly drawn to show information that changes over time.

It is interesting analyzed the *Success Rate* of the launches along the *Years.*

- Success Rate vs. Years

https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/F-EDA_data_visualization.ipynb

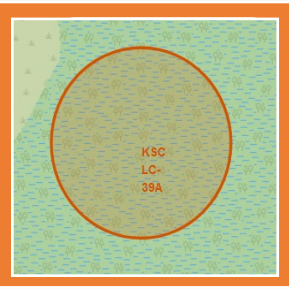
EDA with SQL

SQL queries to gather information about the dataset.

- **All launch sites** in the space mission
- **Launch sites names begin** with **'CCA'** (5 records)
- **Total payload mass carried** by boosters launched by NASA (CRS)
- **Average payload mass carried** by booster version **F9 v1.1**
- **First successful ground landing date**
- **Boosters** with **successful drone ship landing with payload between 4000 and 6000 kg**
- **Total number of successful and failure** mission outcomes
- **Booster versions** which have carried the **maximum payload mass**.
- **2015 records** displaying month, failure outcomes in drone ship, booster versions, launch site
- **Rank landing outcomes between 2010-06-04 and 2017-03-20**, in descending order.

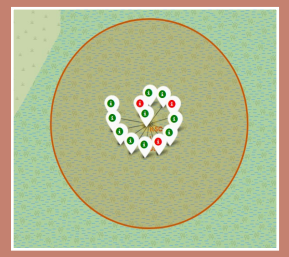
https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/D-EDA-SQL.ipynb

Build an Interactive Map with Folium



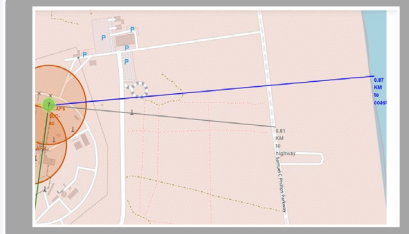
Circles For Launch Sites

- **To visualize the Launch Data into an interactive map**
- Coordinates of each launch site were pointed with a *folium.Circle*
- They were labeled with the name of the launch site with *folium.Marker*



Marker Cluster For Launch Outcomes

- **To visualize the outcomes (failures and success)**
- Coordinates were pointed on the map with a *MarkerCluster()* in pop-up markers
- Assigned failures to class 0 with Red
- Assigned successes to class 1 with Green



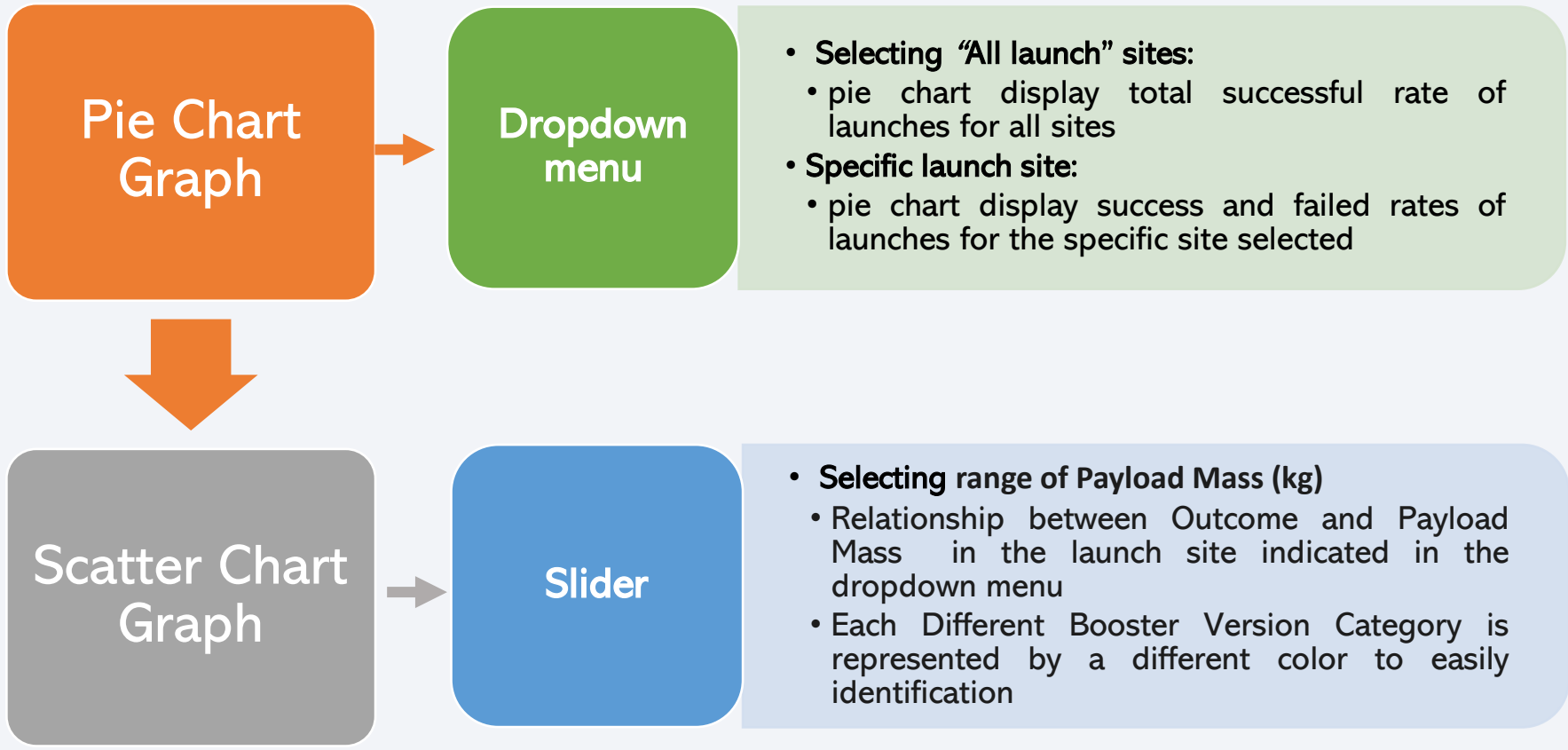
Lines For Distances

- **To visualize the distance from the Launch Site to various landmarks**
- *folium.PolyLine* object were used to join with a line the lauch site and landmark
- They were labeled with the distance using *folium.Marker*
- Haversine's formula were used to calculated the distance

https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/G-Folium_Analysis_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

*See Appendix C for code



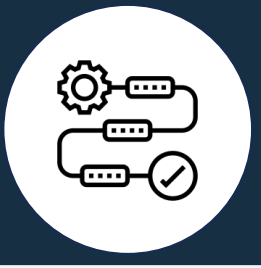
Pie Charts illustrates numerical rates. It is the best option for rates because the area is proportional to the total quantity it represents.

Scatter plots shows relationship between two variables. It is the best method to identify non-linear pattern because:

- *The range of data, can be easily determined.*
- *Observation and reading are straightforward.*

https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/H-Interactive%20Dashboard%20with%20Ploty%20Dash.ipynb

Predictive Analysis (Classification)



MODEL DEVELOPMENT

Model Building

- Identify the predictor variables as X
- Identify the target variable as Y
- Standardize the data in X
- Split the data into training and testing data
- Parameters are optimized by cross-validated grid-search over a parameter grid.
- Fit the model to find the best parameters with training data

Improving Model

- Data normalization (standardization)
- Multiple Algorithms models
  - Logistic Regression
  - SVM
  - Decision Tree
  - KNN Neighbors
- Hyperparameter Tuning (gris-search CV)
- Cross Validation

Model Evaluation

- Calculate the accuracy of the fit modeling to training data with *best_score_* attribute
- Determine test model accuracy with the *.score()* method using testing data
- Evaluates classification accuracy by computing the confusion matrix
- Compare and evaluate classifier's performance with ROC and Precision-Recall curves

Found the Best Model

- Model with the higher test accuracy score
- Model with the higher F1 score, or better confusion matrix
- Model with the best ROC or Precision-Recall curves

https://github.com/EvaTartaruga/SpaceRacewithDataScience/blob/main/l-SpaceX_Machine_Learning_Prediction.ipynb

Results

Section 2: Exploratory data analysis (EDA) results

- Data Visualization through Matplotlib
- Query specific information with SQL

Section 3: Geospatial Visualization results

- Geographical Analysis with Folium

Section 4: Interactive Dashboard with Ploty Dash

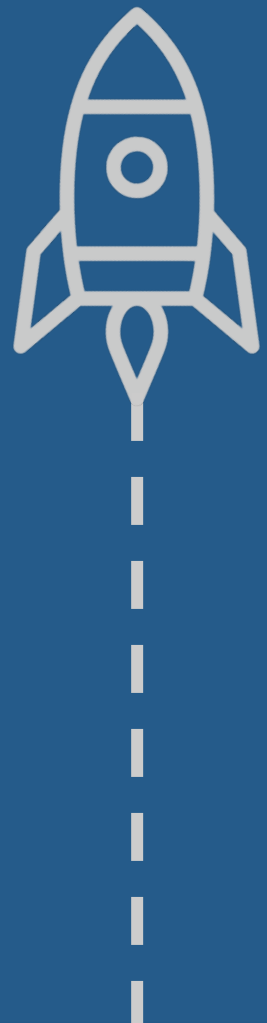
- Successful Rate for Launch Sites
- Correlation with selected Payload by Booster Category

Section 5: Machine Learning Predictive results

- Classification models:
  - Logistic regression, SVM, Regression Tree, KNN Neighbours
- Model accuracy
- Confusion Matrix

Section 2: Data Results

Insights drawn from EDA

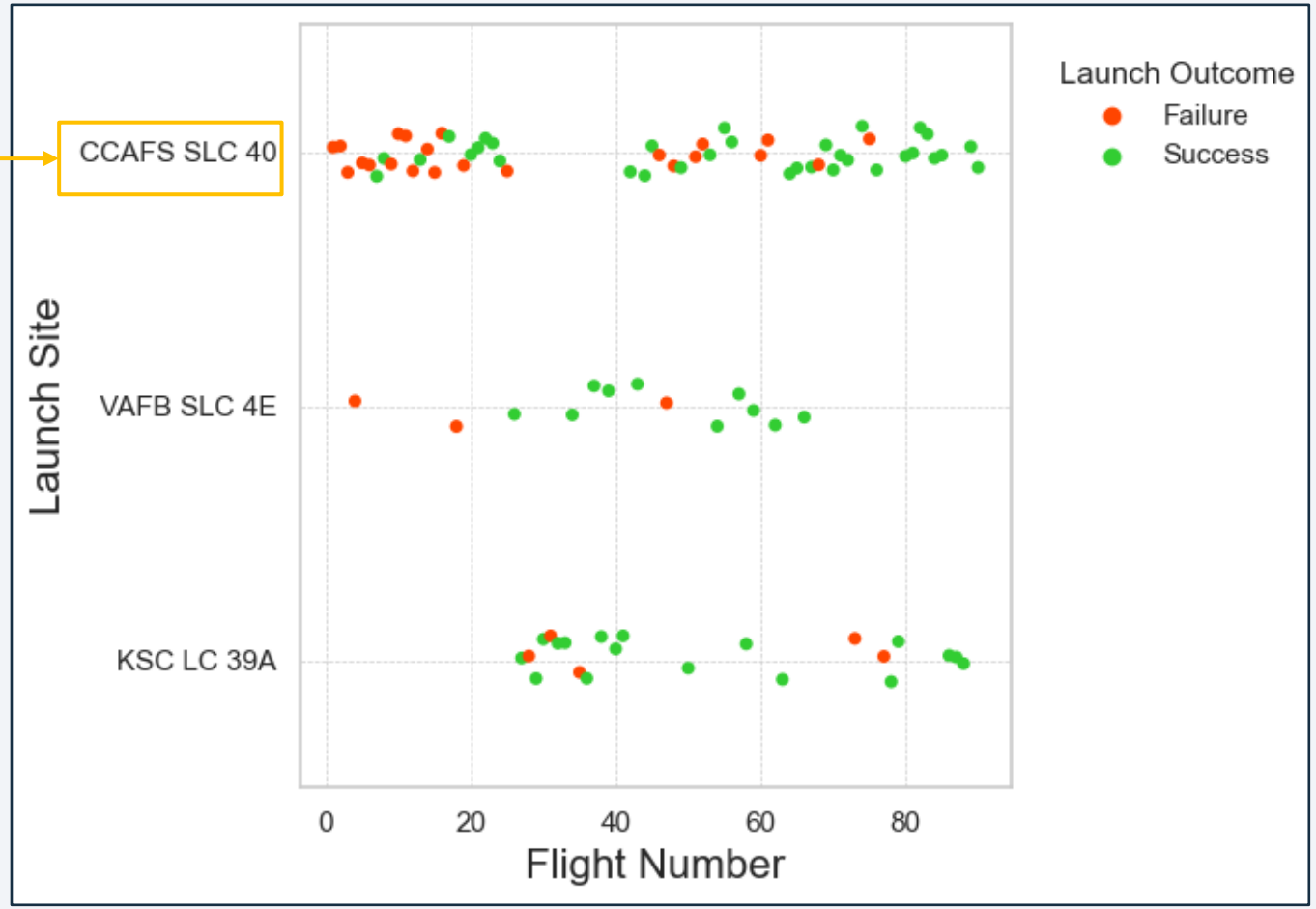


Flight Number vs. Launch Site

More flight attempts.
Flight number < 80 Success Rate low

Maybe this place it is used as testing site.
The modifications were applied to the other sites to check success, so it is the reason of high Rate of Success for the other two sites*

Successful outcome increase with the flight number

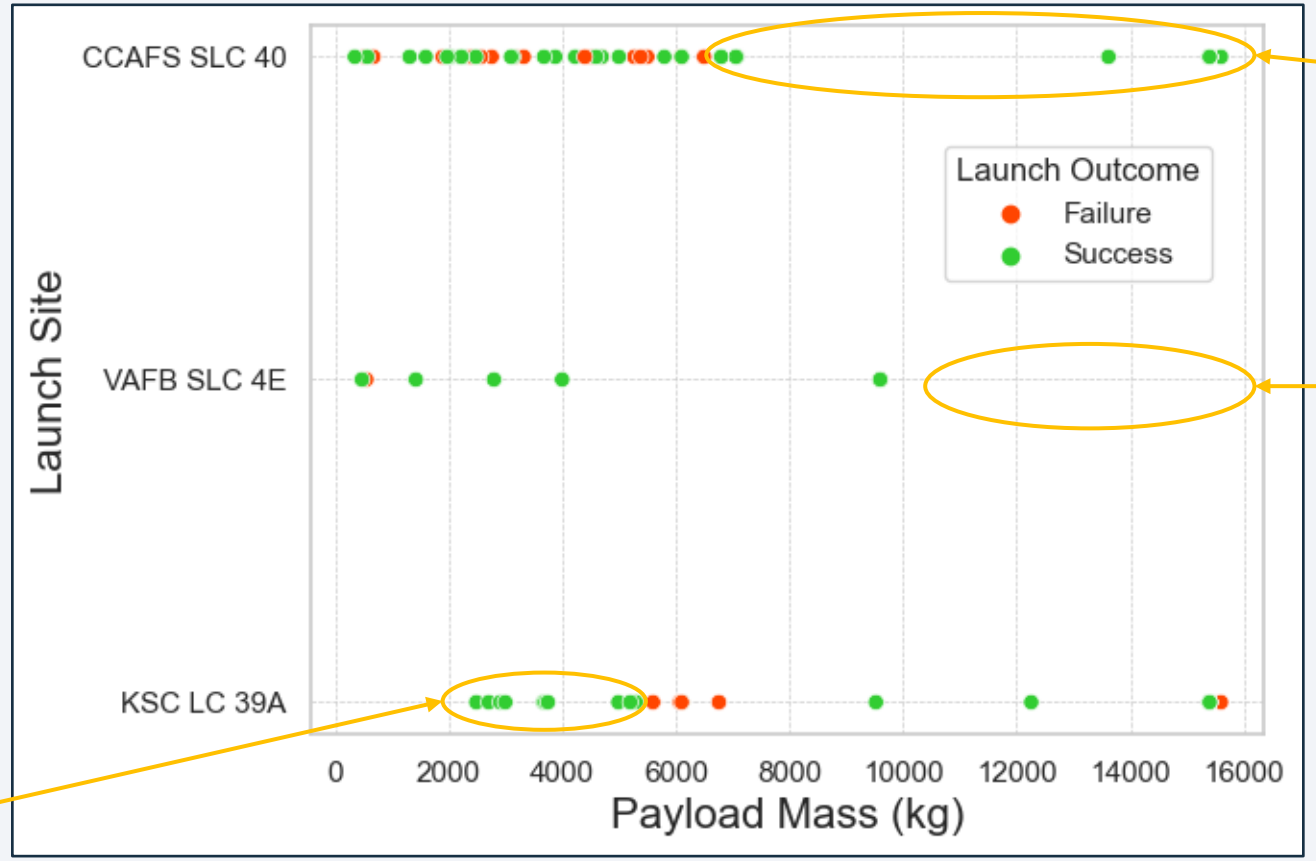


Flight Number: the continuous launch attempts
Launch Sites: code of location of Space X facilities

*See Appendix D for values

Payload vs. Launch Site

Successful outcome seems to be correlated with Payload & Launch Site



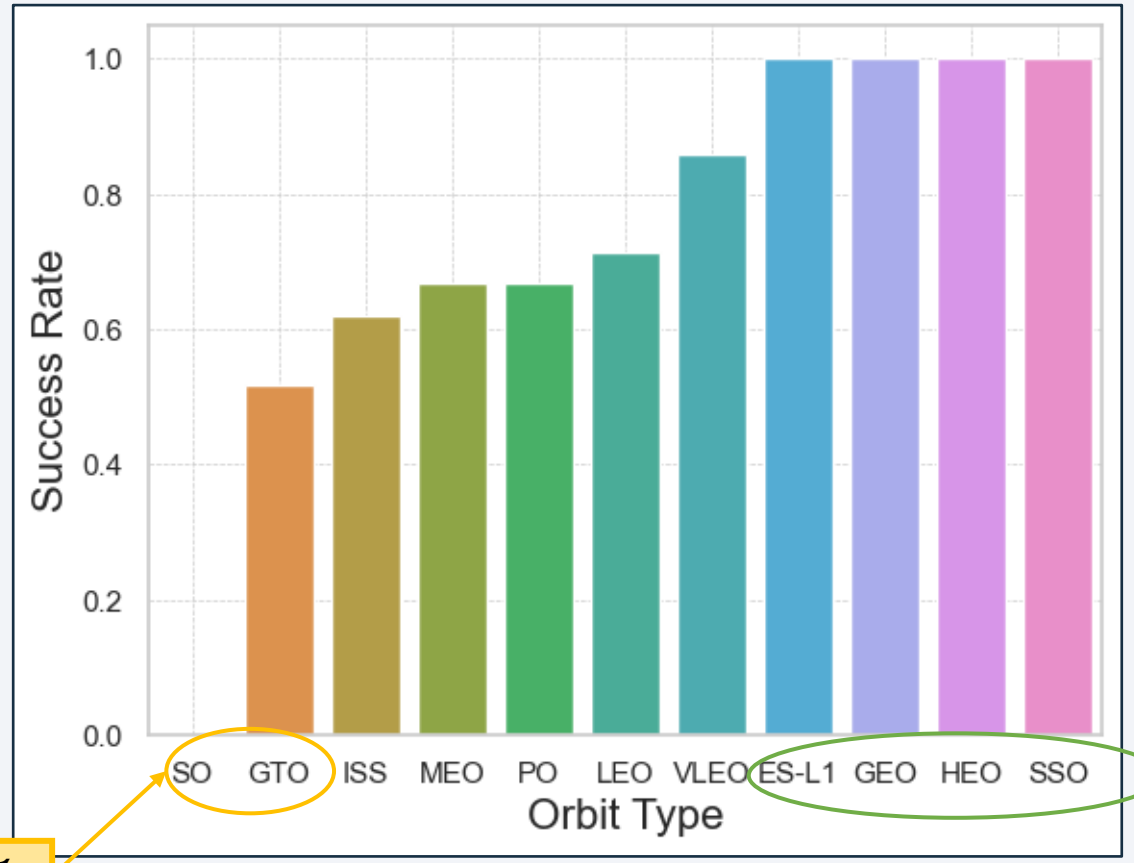
High Success Rate for heavy Payloads in CCAFS

No heavy Payloads (> 10000 kg) in VAFB site

High Success Rate for light Payloads in KSC

Success Rate vs. Orbit Type

*See Appendix E for more info



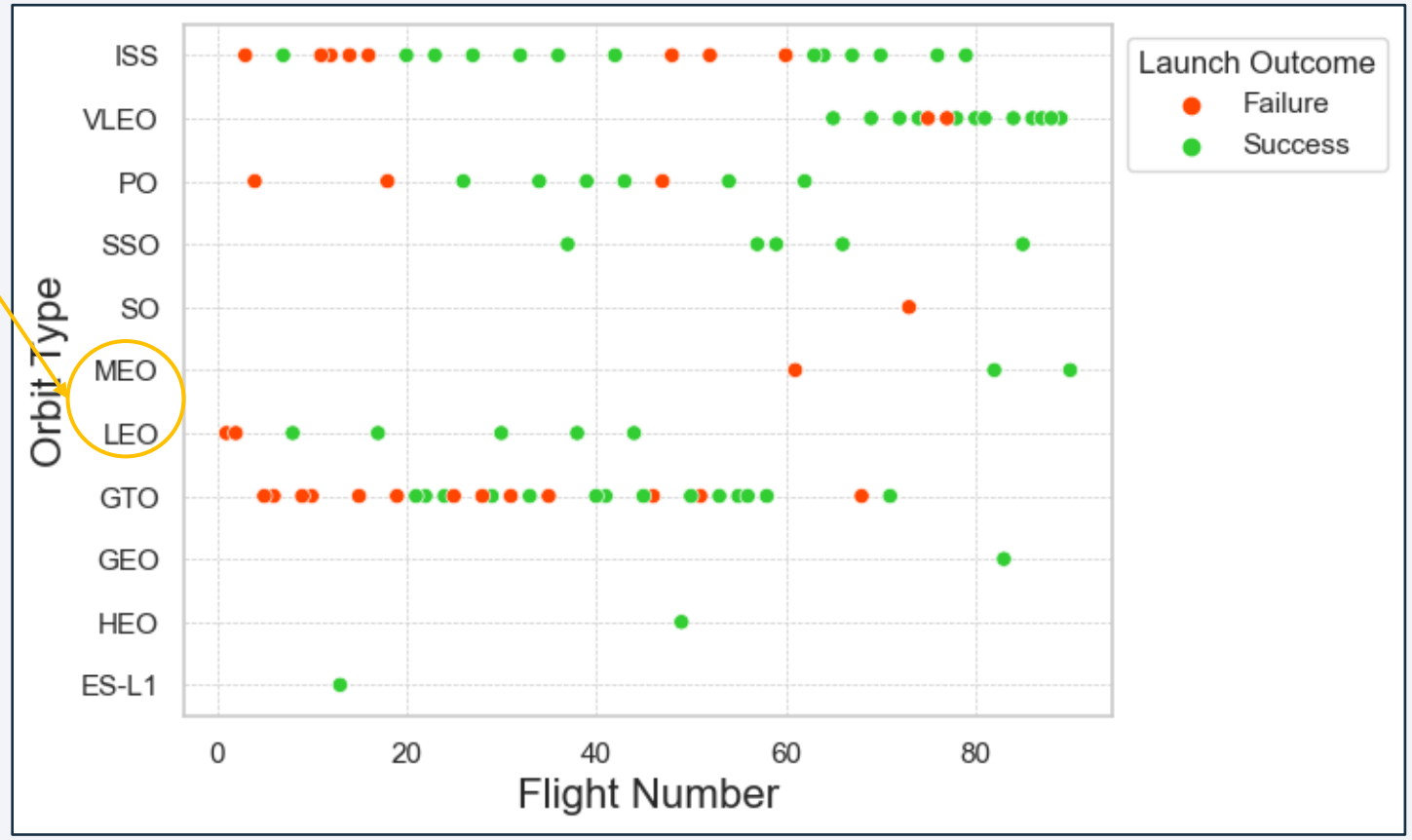
Success Rate < 50%

Success Rate = 100%

Flight Number vs. Orbit Type

For LEO and MEO orbits the Success Rate appears related to the number of flights

Low height orbits (ISS, VLEO, PO) and GTO show more number of flights, maybe to be the more accessible orbits

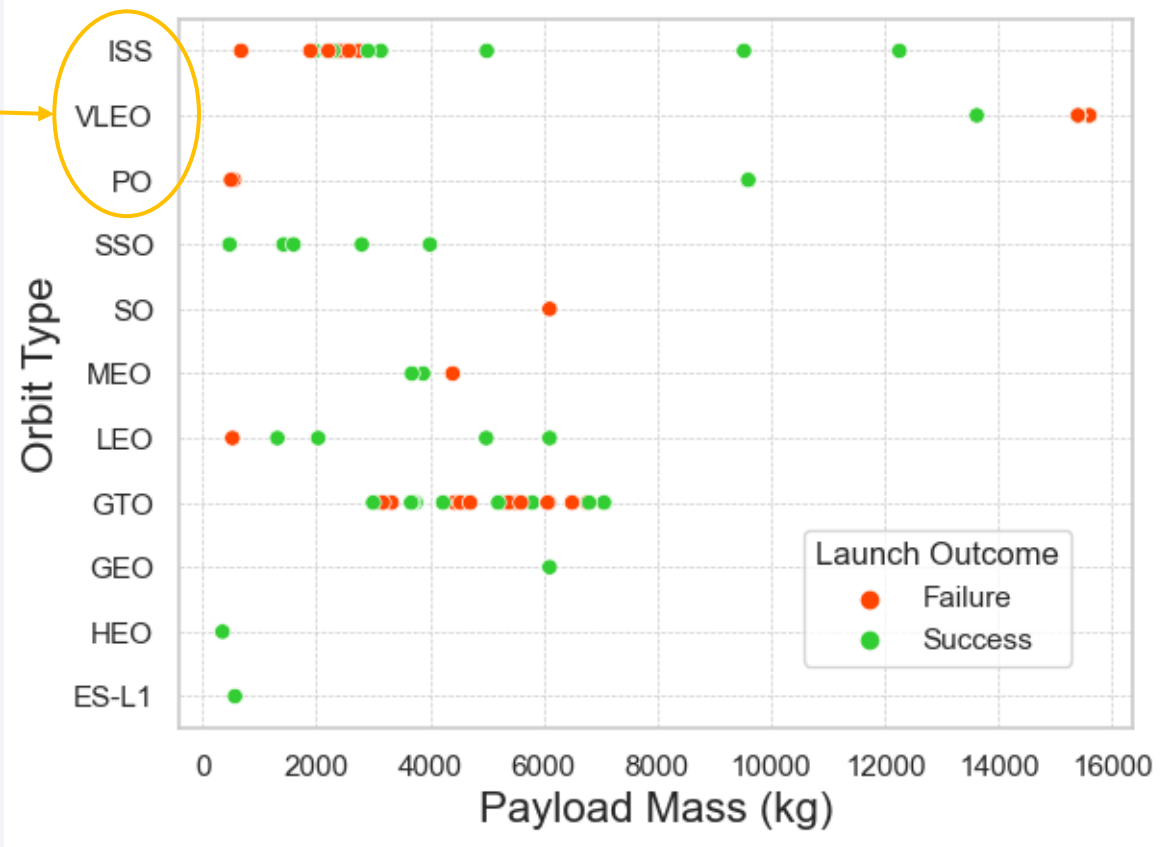


There seems to be no clear relationship between Flight Number and Orbit Type, specially for GTO

Payload vs. Orbit Type

Heavy Payloads (>10000 kg) are used only in **ISS**, **VLEO** and **PO**, low height orbits

	Orbit	km
4	ISS	408
10	VLEO	450
7	PO	600
8	SO	700
9	SSO	700
6	MEO	2000
5	LEO	2000
2	GTO	35786
1	GEO	35786
3	HEO	35786
0	ES-L1	1500000



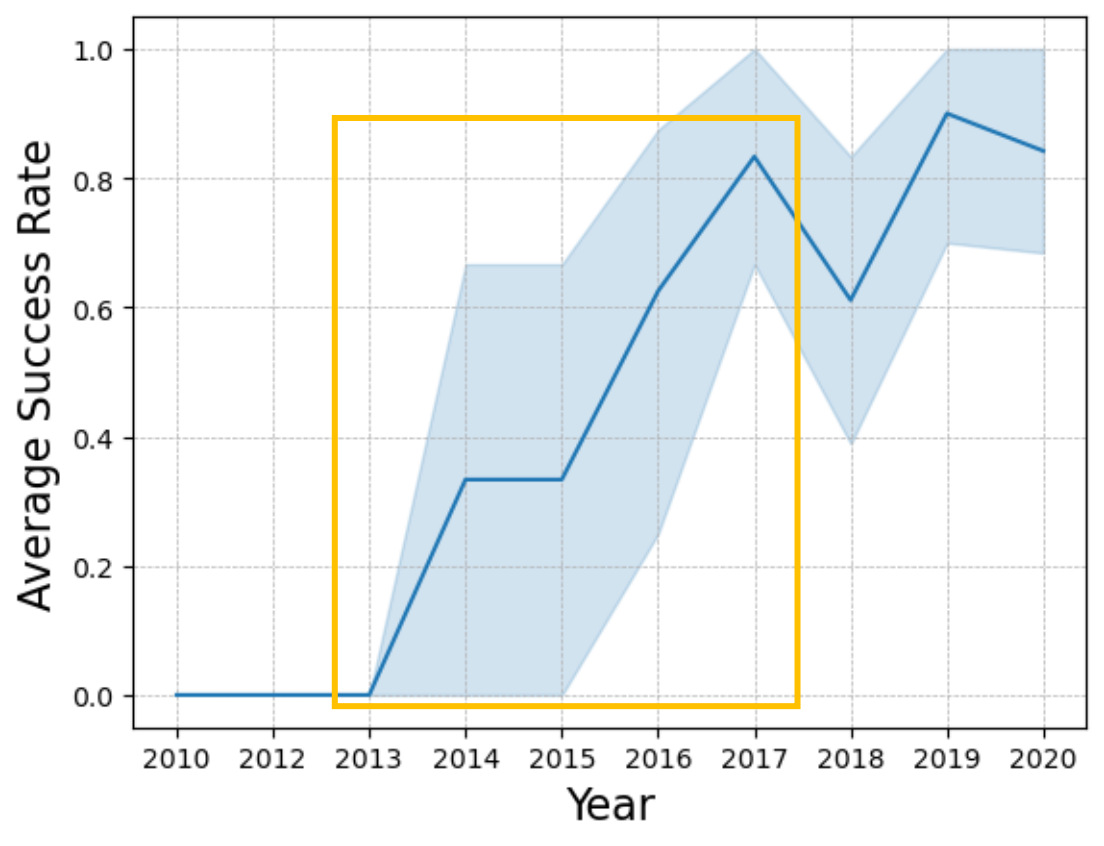
<u>With heavy payloads:</u>
High successful landing are more for **Polar**, and **ISS**.

<u>With light payloads:</u>
High Successful landing are more for **SSO, LEO**

GTO have intermediate successful landing

GEO, HEO, ES-L1, have success outcomes, but only one attempt, but are further.

Launch Success Yearly Trend



Success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing.

All Launch Site Names

SQL Query:

```
3  Select distinct Launch_Site from spacextable
```

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

Using the word DISTINCT in the query means that it will only show Unique values in the *Launch_Site* column of table "spacextable" in the database.

4 Launch Sites location of Space X facilities were found

Launch Site Names Begin with 'CCA'

SQL Query:

```
3  select * from spacextable where Launch_Site like 'CCA%' limit 5
```

Word LIMIT 5 in the query means that it will only show 5 records.

LIKE keyword filter the words with 'CCA%' (% in the end suggests that the *Launch_Site* name must start with CCA).

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

SQL Query:

```
select sum(PAYLOAD_MASS__KG_) from spacextable where Customer='NASA (CRS)'
```

sum(PAYLOAD_MASS__KG_)
45596

The total mass used by NASA (CRS) is **45596 kg**

The function SUM summates the total in the column *PAYLOAD_MASS_KG_*

The WHERE clause filters the dataset to only perform calculations on "*Customer NASA (CRS)*"

Average Payload Mass by F9 v1.1

SQL Query:

```
select avg(PAYLOAD_MASS__KG_) from spacextable where Booster_Version='F9 v1.1'
```

avg(PAYLOAD_MASS__KG_)
2928.4

The function AVG works out the average in the column *PAYLOAD_MASS_KG_*

The WHERE clause filters the dataset to only perform calculations on *Booster_version F9 v1.1*

The mass used in booster F9 v1.1 is **2928 kg** in average per booster

First Successful Ground Landing Date

SQL Query:

```
select min(Date) from spacextable where Landing_Outcome='Success (ground pad)'
```

min(Date)
2015-12-22

The first successful ground landing was in **2015-12-22**

The function MIN works out the minimum date in the column *Date*

The WHERE clause filters the dataset to only perform calculations on *Landing_Outcome* *"Success (ground pad)"*

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query:

```
select Booster_Version from spacextable where (
    Landing_Outcome='Success (drone ship)' and (PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000 )
)
```

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

Selecting *Booster_Version* to show only the results of this column

The WHERE clause filters the dataset *Landing_Outcome*

Using the AND keyword specifies another filter conditions for Payload_MASS_KG to be between 4000 and 6000

Only **4 booster versions** accomplish the successful landing in drone ship with payloads between 4000 and 6000 kg.

Total Number of Successful and Failure Mission Outcomes

SQL Query:

```
SELECT (select count(Mission_Outcome) from spacextable where Mission_Outcome LIKE '%Success%') as Success,
       (select count(Mission_Outcome) from spacextable where Mission_Outcome LIKE '%Failure%') as Fail;
```

Success	Fail
100	1

The **successful** mission outcome was **100 in total** (99.01%)
The **failure** in the mission was low, **1 in total** (0.09%)

Here, subqueries were used to produce the results.

The LIKE '%Success%' or '%Failure%' wildcard shows that in the record the *"Success"* or *"Failure"* word is in any part of the string in the records.

The AS keyword rename the output query for clarification.

Boosters Carried Maximum Payload

SQL Query:

```
select Booster_Version from spacextable where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextable);
```

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

```
1 %sql select max(PAYLOAD_MASS__KG_) from spacextable
 * sqlite:///my_data1.db
Done.
```

max(PAYLOAD_MASS__KG_)
15600

There are **12 booster Versions** carrying the maximum payload (15600 kg)

Here, a subquery in the WHERE clause was used to produce the results

The subquery filter the *PAYLOAD_MASS_KG_* dataset with the result of apply the function MAX in *PAYLOAD_MASS_KG_* of the spacextable database

MAX(PAYLOAD_MASS_KG_) return the maximum value of the dataset

2015 Launch Records

SQL Query:

```
select  substr(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from spacextable
where (Landing_Outcome='Failure (drone ship)') and (substr(Date,0,5)='2015');
```

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

There are **only 2 results (booster version B1012 and B1015) in 2015 (January and April)** where the drone ship landing was fail, both in the same location

As SQLLite does not support month names, was used substr(Date, 6,2) renamed with AS keyword to get the months and substr(Date,0,5)='2015' for selecting the year.

Selecting the required columns *month*, *Landing_Outcome*, *Booster_Version* and *Launch_site* datasets.

The WHERE clause filters the dataset to only perform *"Failure (drone ship)"* for *Landing_Outcome* and the year 2015, using the AND keyword specifying both filter conditions.

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

SQL Query:

```
select Landing_Outcome, count(Landing_Outcome) as Count_of_Landings from spacextable
where Date between '2010-06-04' and '2017-03-20'
group by Landing_Outcome
order by Count_of_Landings desc;
```

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

Function COUNT counts records in column *Landing_Outcome* and the result is renamed with the keyword AS.

WHERE filters data between the dates selected and joined with the AND word.

GROUP BY is used to group the *Landing_Outcome* values.

DESC means its arranging the dataset into descending order.

The most number of landings (excluding no attempt) were in drone ship between 2010-06-04 and 2017-03-20

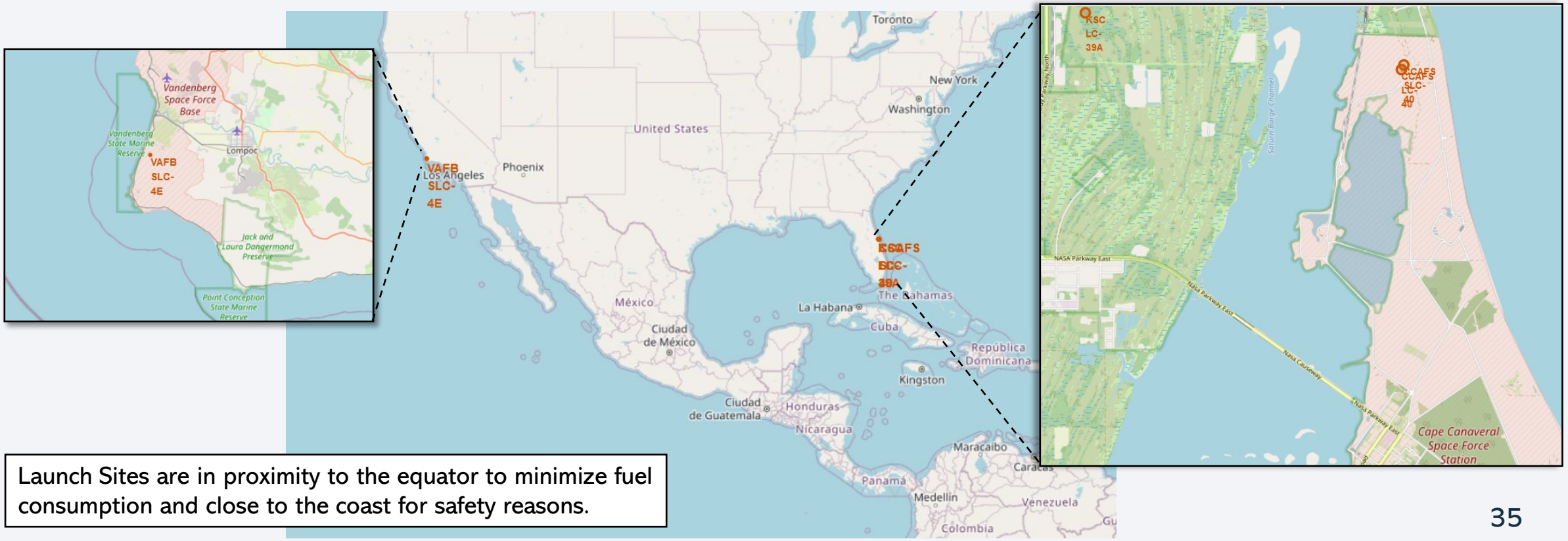
Section 3: Geographical Analysis Results

Launch Sites
Launch Outcomes
Proximities Analysis

Launch Site Map Locations

SpaceX launch sites are in the United States of America coasts inside installations of the United States Space Force's.

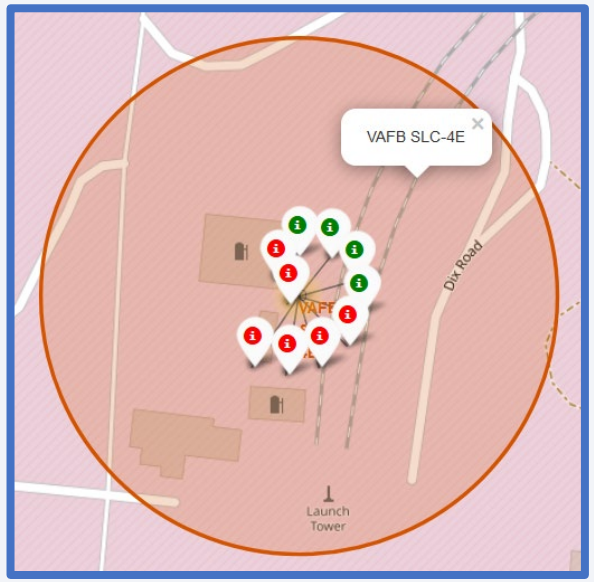
- VAFB SLC- 4E is placed in California coast inside Vandenberg Space Force Base
- KSC LC-33A, CAFS-SLC-40 and CAFS-LC-40 are situated in Florida coast in the Cape Canaveral Space Force Station



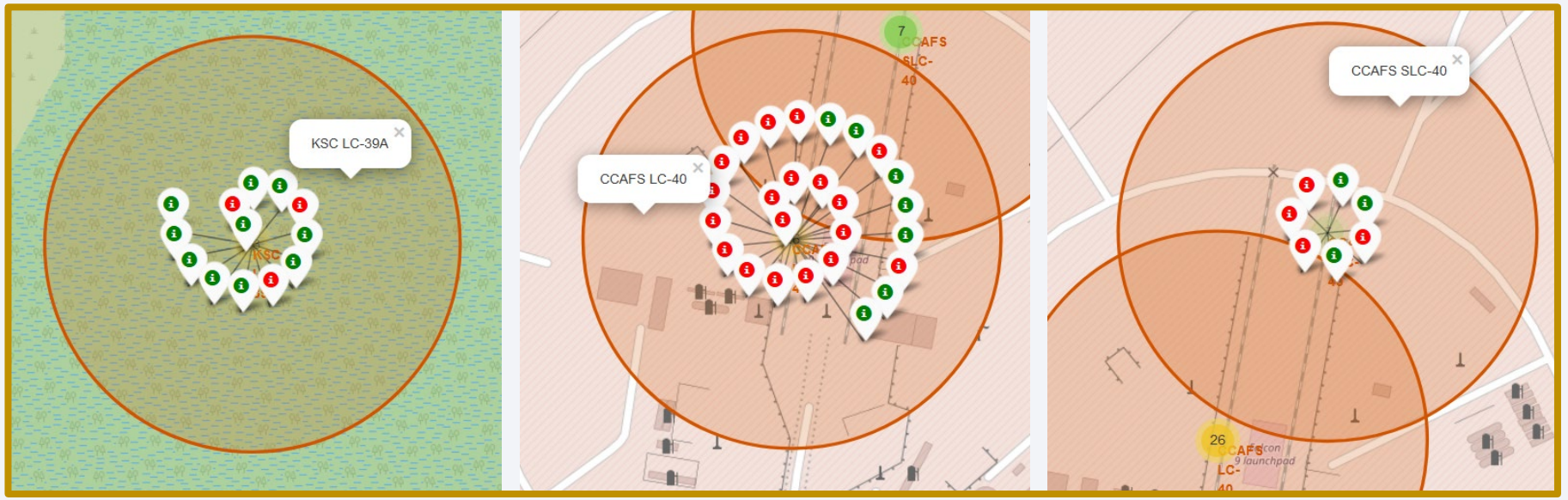
Launch Sites are in proximity to the equator to minimize fuel consumption and close to the coast for safety reasons.

Success/Failed Launches Outcomes

California Launch Site



Florida Launch Sites



Number of markers proportional of number of flights.

- Green Marker shows successful **Launch** Outcome
- Red Markers shows Failures

Launch Site – Landmark Proximity

Launch sites are in **close proximity to:**

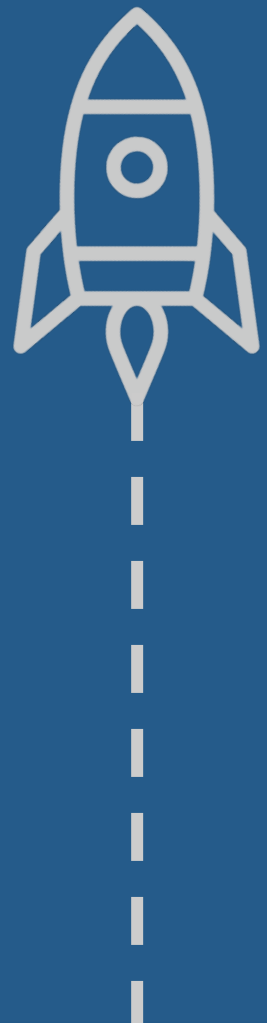
- **highways,** which allows for easy accessibility and easy transportation of materials
- **railways,** which allows transport of materials, specially for heavy cargo
- **coastline,** for at least two safety reasons, rockets can fly over the ocean:
  - to abort launch and attempt water landing and/or to failure testing security
  - to minimize people and property at risk from falling debris

Launch sites are **far for:**

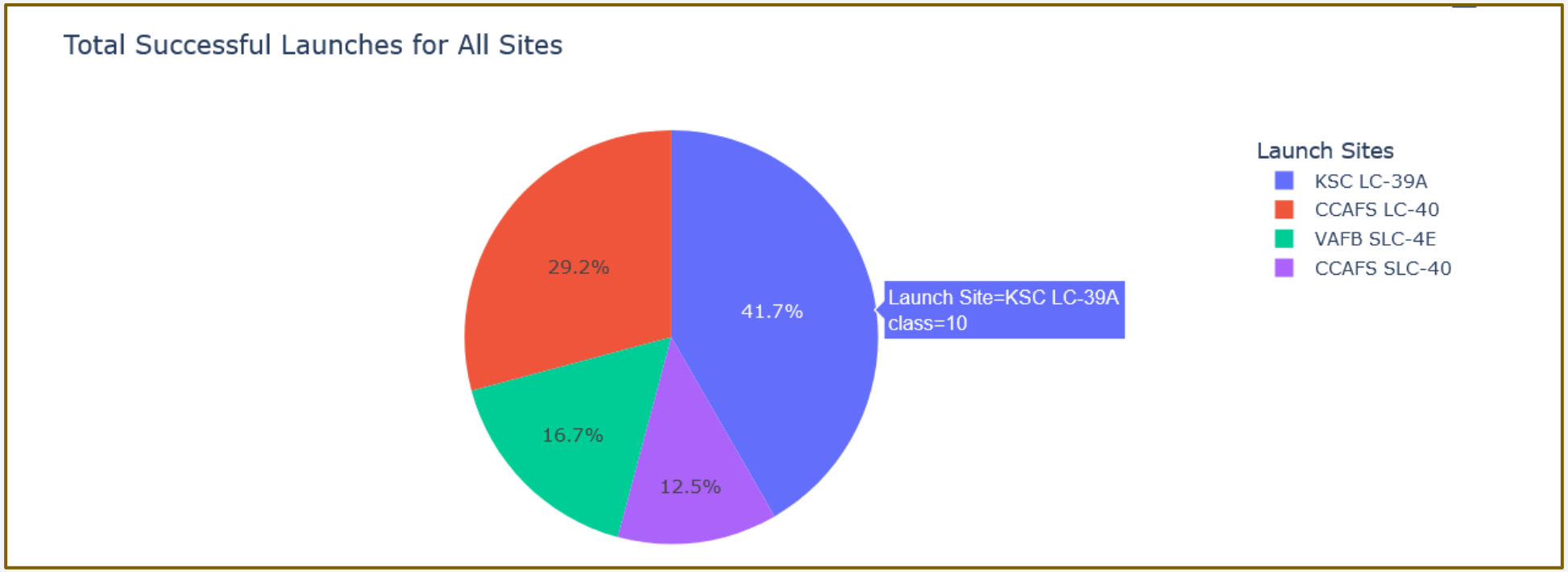
- **cities,** which minimizes danger to population areas

Section 4: Interactive Insights

Dashboard Presentation
using Plotly Dash



Successful Rate by Launch Site



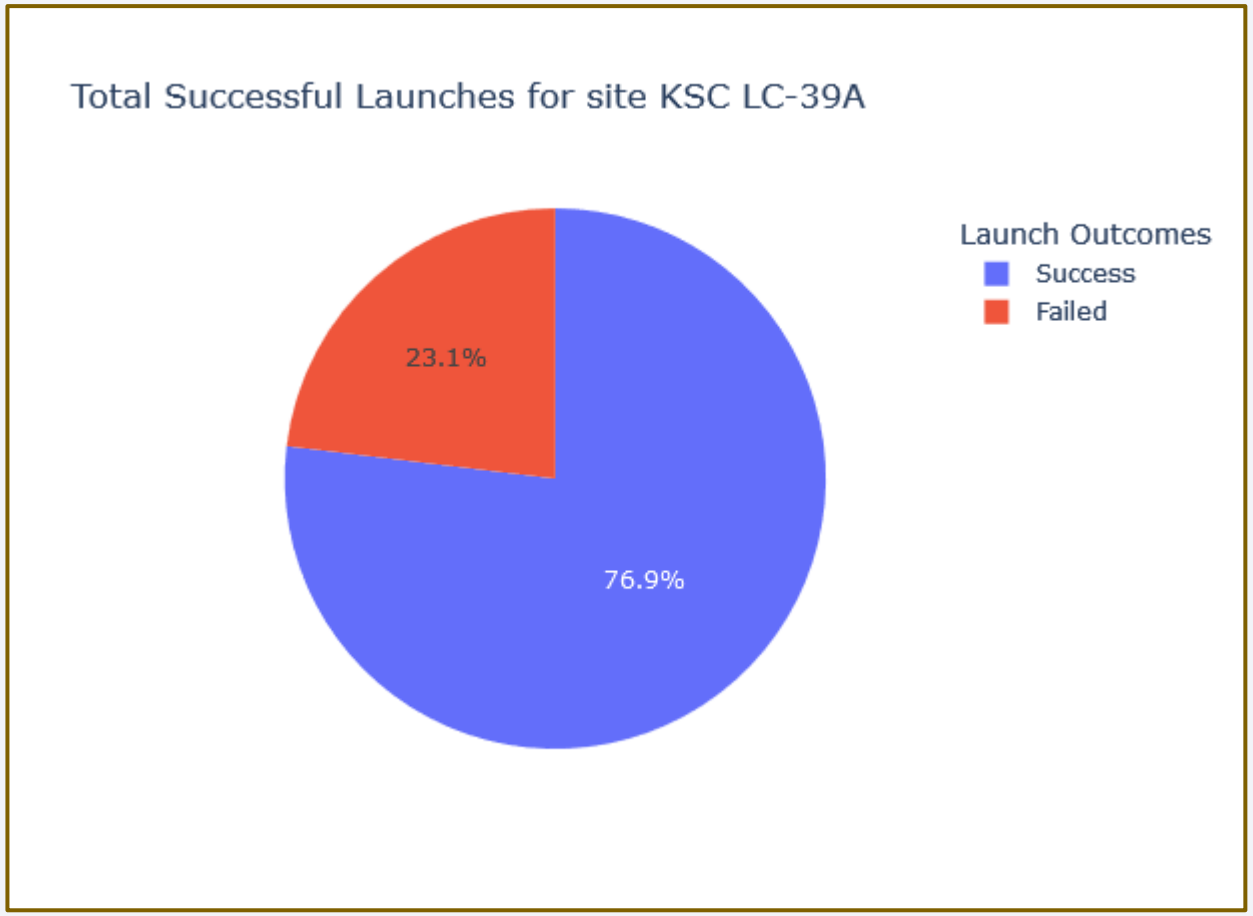
KSC LC-39A launch site:

- Highest successful Rate

- Largest successful launches*, (observed interactively with the pop-up)

*See Appendix F for all the pop-up information

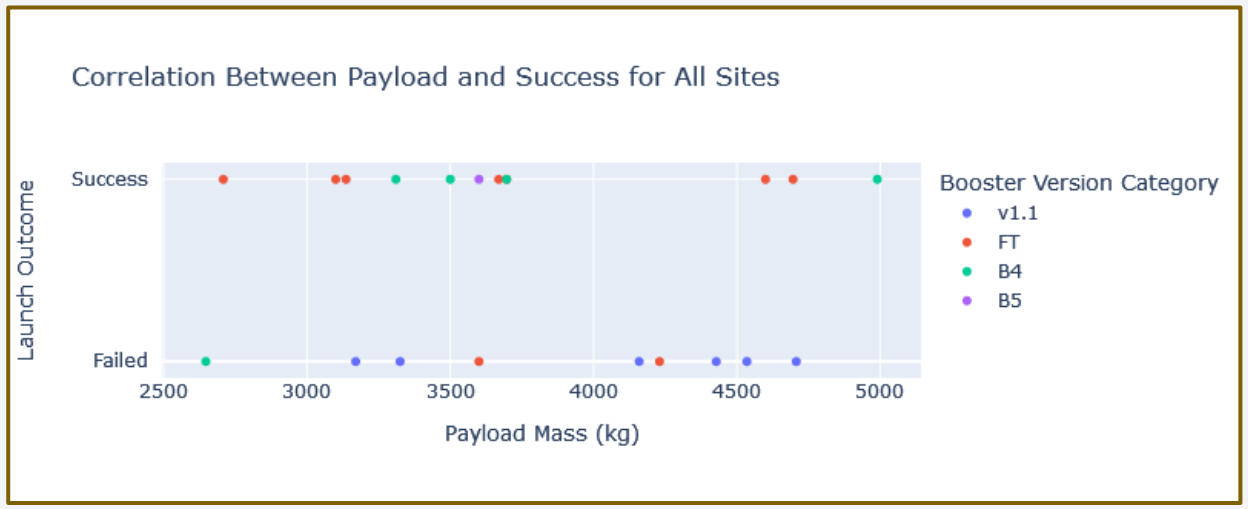
KSC LC-39A Launch Outcome Rates

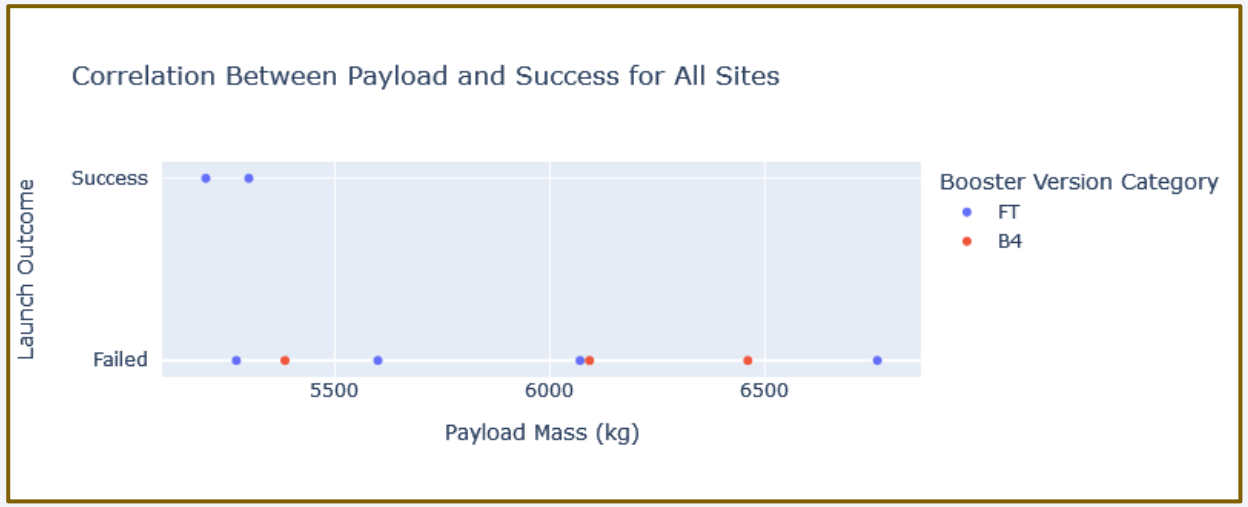


Specifically, KSC LC-39A achieved:

- 76.9% success rate
- 23.1% failure rate

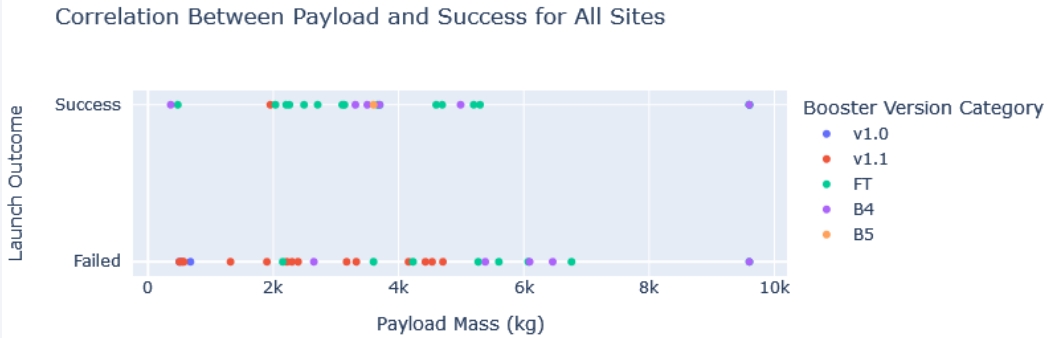
Payload Mass vs. Launch Outcome





- Payload from **2500 to 5000 kg**, have a **122% success** rate
- Payload from **5000 to 7500 kg** have a **29% success** rate
- Thus, Falcon9 have positive outcomes for Payloads between 2500 and 5000 kg
- **FT booster version** has the **highest launch success rate** 12 success and 7 fails **from 2500 kg to 7000 kg**, 63% launch success rate (excluding B5 only with 1 attempt).

*

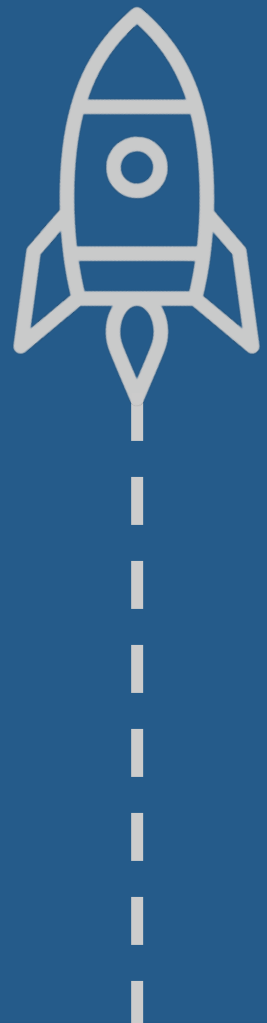


* This screenshot is displayed for comparison purposes

** See Appendix G for KSC LC-39A graph information

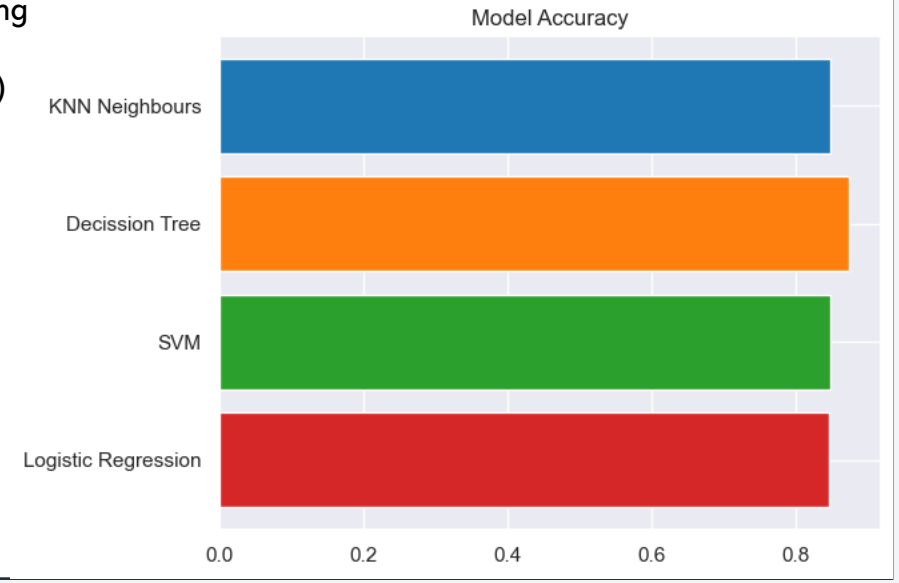
Section 5: Machine Learning Results

Predictive Analysis
(Classification)

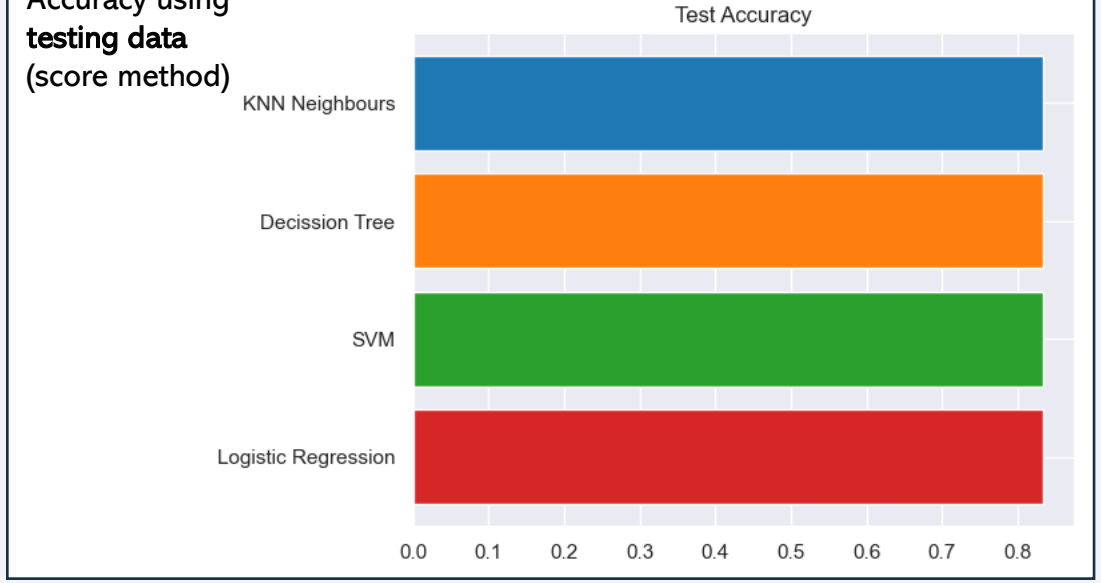


Classification Accuracy

Accuracy using **training data** (best_score_)



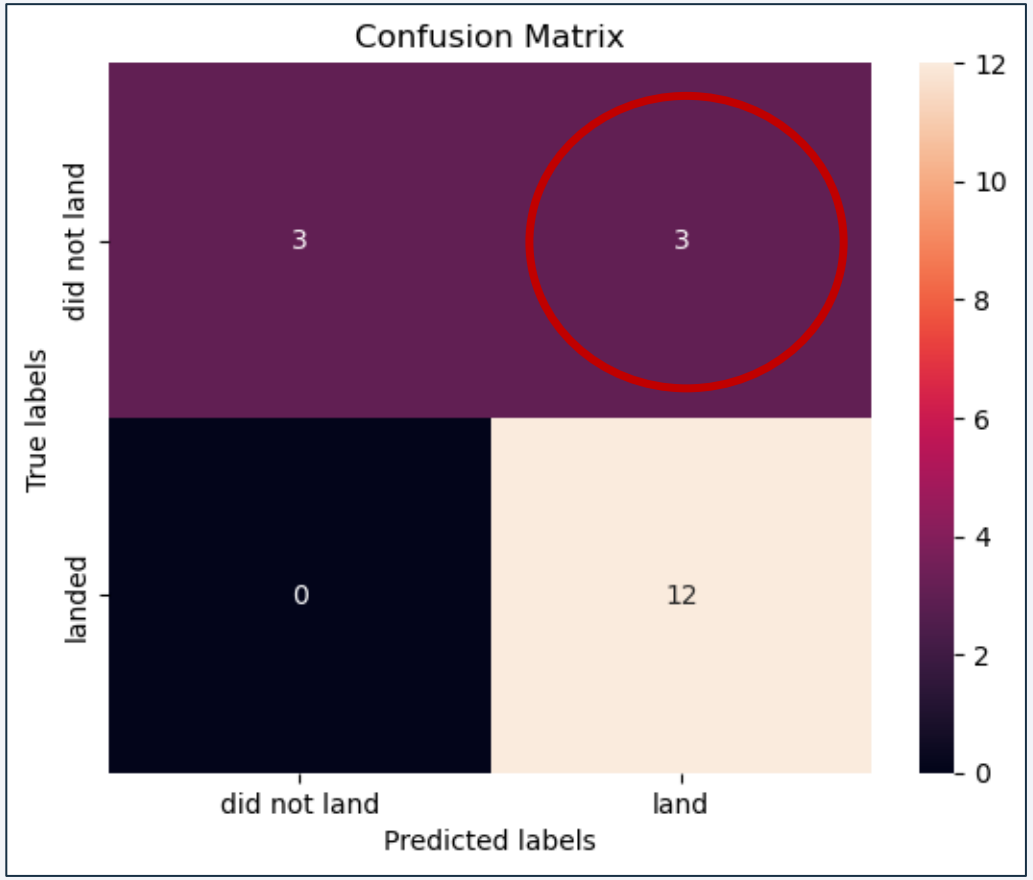
Accuracy using **testing data** (score method)



- Test Accuracy: all ML classifiers preformed practically the same accuracy for testing data. **No model has the highest precision, it is the same for all. A new evaluation method is required to choose a model.**
- The model accuracy: decision tree fit train data slightly better but test data same, so maybe it overfit.
- SVM and KNN Neighbours displays the same best test model accuracy after decision tree

Model	Model Accuracy	Test Accuracy	F1 Score
Logistic Regression	0.846429	0.833333	0.814815
SVM	0.848214	0.833333	0.814815
Decission Tree	0.875000	0.833333	0.814815
KNN Neighbours	0.848214	0.833333	0.814815

Confusion Matrix



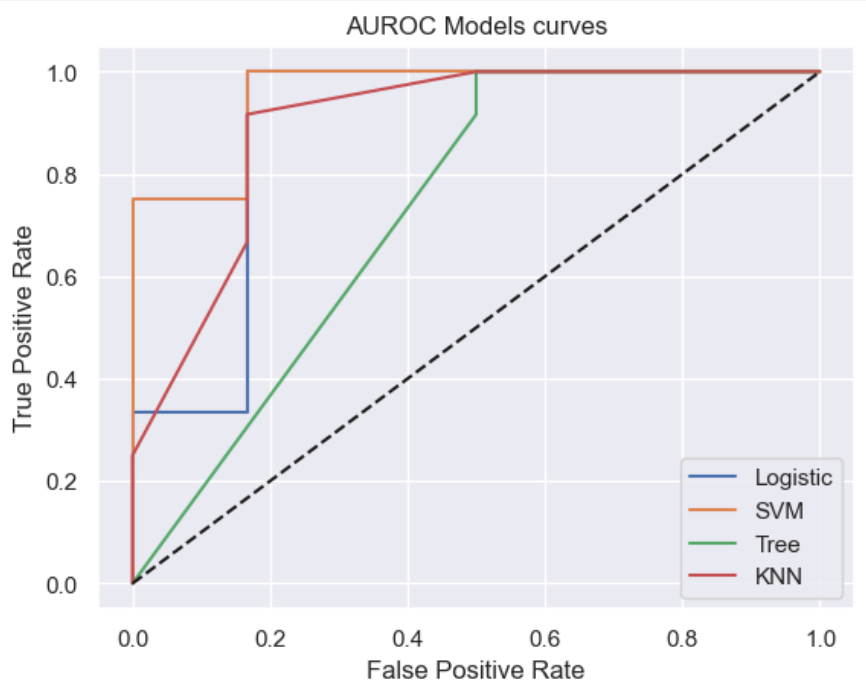
Confusion Matrix of Decission Tree

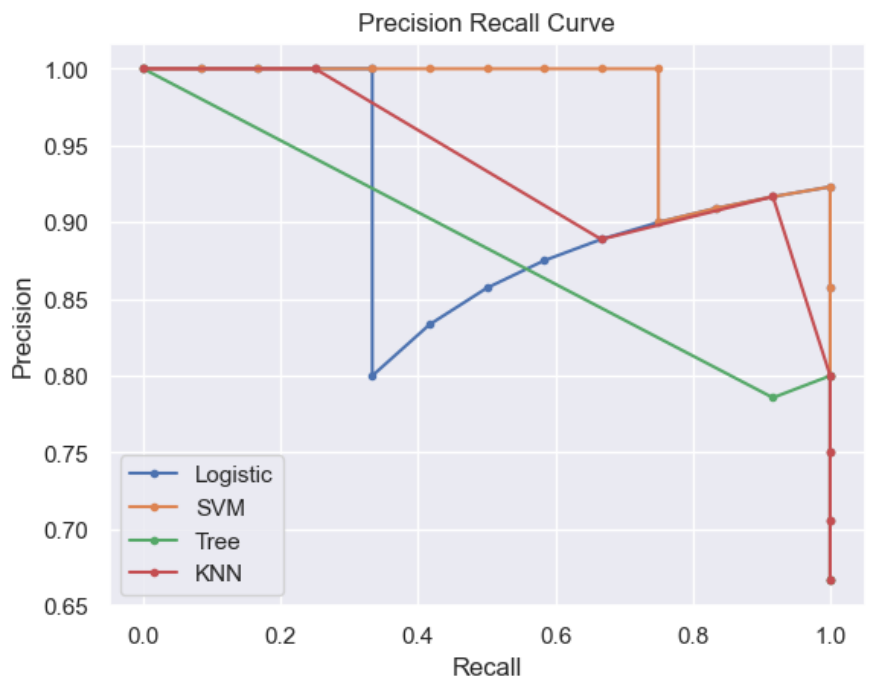
* See Appendix H

		Predicted Values (Test result)	
		Predicted negative	Predicted positive
Actual values	Negative (N)	True Negative (TN) Correctly predicted as 0	False Positive (FP) Type I error Overestimation Incorrectly predicted as 1
	Positive (P)	False Negative (FN) Underestimation Type II error Incorrectly predicted as 0	True Positive (TP) Correctly predicted as 1

- Model **correctly classify all success outcomes** (class 1)
- Model **fails to classify negative outcomes** (class 0)
- The major problem is false positives (Overestimation)
- All models show the same confusion matrix*

AUROC & Precision-Recall Curve





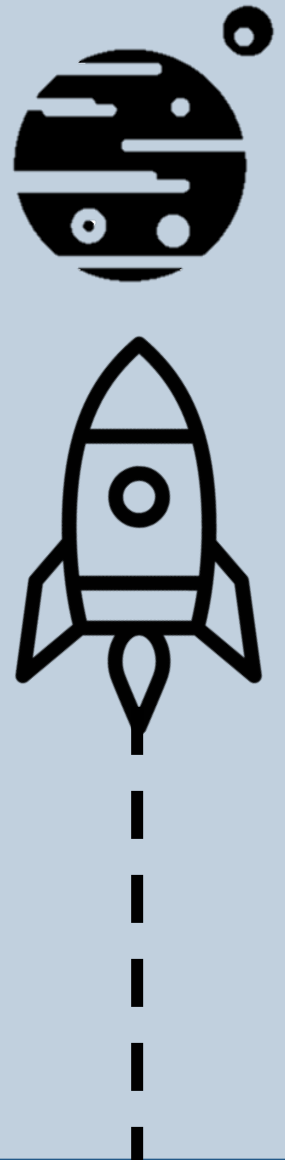
* See Appendix I for further information

Model	Model Accuracy	Test Accuracy	F1 Score	AUROC	AUPRC
Logistic Regression	0.846429	0.833333	0.814815	0.89	0.925267
SVM	0.848214	0.833333	0.814815	0.96	0.979070
Decission Tree	0.875000	0.833333	0.814815	0.73	0.786905
KNN Neighbours	0.848214	0.833333	0.814815	0.90	0.916204

- AUROC and Precission-Recall metrics allows to distinguish the best classifier
- **SVM have the highest AUROC and AUPRC**
- These metrics confirms the overfitting of Decision Tree model

Conclusions

- The success rates for SpaceX launches is directly proportional time in years and number of flights, since they will eventually perfect the launches.
- Rockets, specially with FT booster, has the highest launch success for low payloads (<5500 kg), independently of launch site.
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate, but not correlation was found with their distance from Earth. It seems the success is because they are the most common and studied ones.
- KSC LC-39A had the most successful launches from all the sites. All sites are far from cities but close to ocean for safety reasons, and near railways and highways for transportation and accessibility
- SVC Classifier is the best for this data set to predict the launch outcome, according ROC and Precision-Recall curves.



Appendix A

Data Collection – SpaceX API Python Code

1. Getting Response from API

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

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

2. Converting the json file Response to a data frame

```
# Use json_normalize meethod to convert the json result into a dataframe

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

3. Extract Information using ID numbers in the launch data

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getCoreData
getCoreData(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

4. Assign list to dictionary

```
# Create a data from launch_dict
launch_dictionary_df = pd.DataFrame(launch_dict)
```

5. Filter data frame to only include Falcon 9

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=launch_dictionary_df[launch_dictionary_df['BoosterVersion']!='Falcon 1']
data_falcon9['BoosterVersion'].unique()
```

6. Deal with Missing Payload values

```
# Calculate the mean value of PayloadMass column
payload_mean=data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payload_mean, inplace=True)
data_falcon9.isnull().sum()
```

7. Export to csv file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Appendix B

Data Collection – Scraping Python Code

1. Perform an HTTP GET to request the data

```
# use requests.get() method with the provided static_url
# assign the response to a object
response=requests.get(static_url)
```

2. Creating the Beautiful Soup Object

```
# Use the find_all function in the BeautifulSoup object,
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

3. Finding the HTML tables

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response
soup = BeautifulSoup(response.text, 'html.parser')
```

4. Extract column names from tables *

```
extracted_row = 0

#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
```

5. Parsed launch HTML tables into dictionary

```
th_elements=first_launch_table.find_all('th')

for i in th_elements:
    column_name=extract_column_from_header(i)

    if column_name!=None and len(column_name)>0:
        column_names.append(column_name)
```

6. Convert dictionary to data frame

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

7. Export to csv file

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


* See notebook for complete code

Appendix C: Dashboard in Jupyter Notebook

- Rather than using the provided *theiadocker* instance that does not persist data, I am adapting the assignment to be run on Jupyter notebook. I hope you enjoy creating your own Dash app in Jupyter Notebook.

- These are the required libraries

```
# Import required packages
import pandas as pd
import plotly.express as px
import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output

import wget
import webbrowser
```

- The skeleton Dash app is the same as the *theiadocker*

```
# skeleton Dash app to be completed in this lab, downloaded from:
# https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/


# Create a dash application
app = dash.Dash(__name__)


# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),

                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                html.Br(),
                                html.Div([
                                        html.Div(dcc.Dropdown(id='site-dropdown',
```

- To visualize the dashboard we can choose between notebook or web-browser visualization

```
# Run the app
if __name__ == '__main__':
    app.run_server(port=8002, host='127.0.0.1') # to open in the notebook
    webbrowser.open_new('http://127.0.0.1:8002') ## to open in a new tab in the webbrowser
```

* Check the link of the notebook to see the complete code

Appendix D

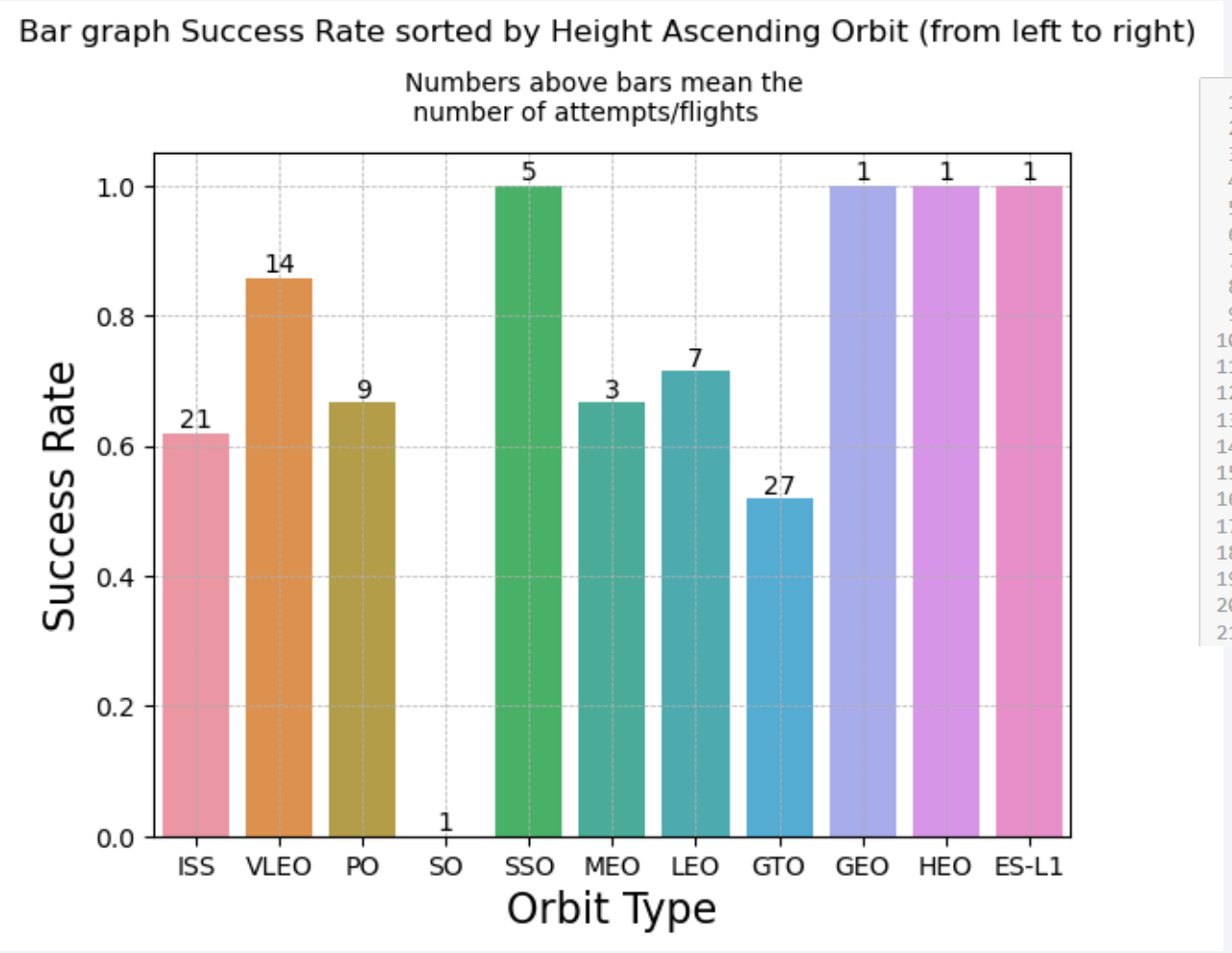
Success Rate by Launch Site

In [80]:

```
# calculate success rate of each site
site=df['LaunchSite'].value_counts()
success = df['LaunchSite'][(df['Class'] == 1)].value_counts()

rate=success*100/site

for i in range(len(rate)):
    print(f"Success rate at {site.index[i]}: {rate.iloc[i]:.2f}%")

```

```
Success rate at CCAFS SLC 40: 60.00%
Success rate at KSC LC 39A: 77.27%
Success rate at VAFB SLC 4E: 76.92%
```

Appendix E

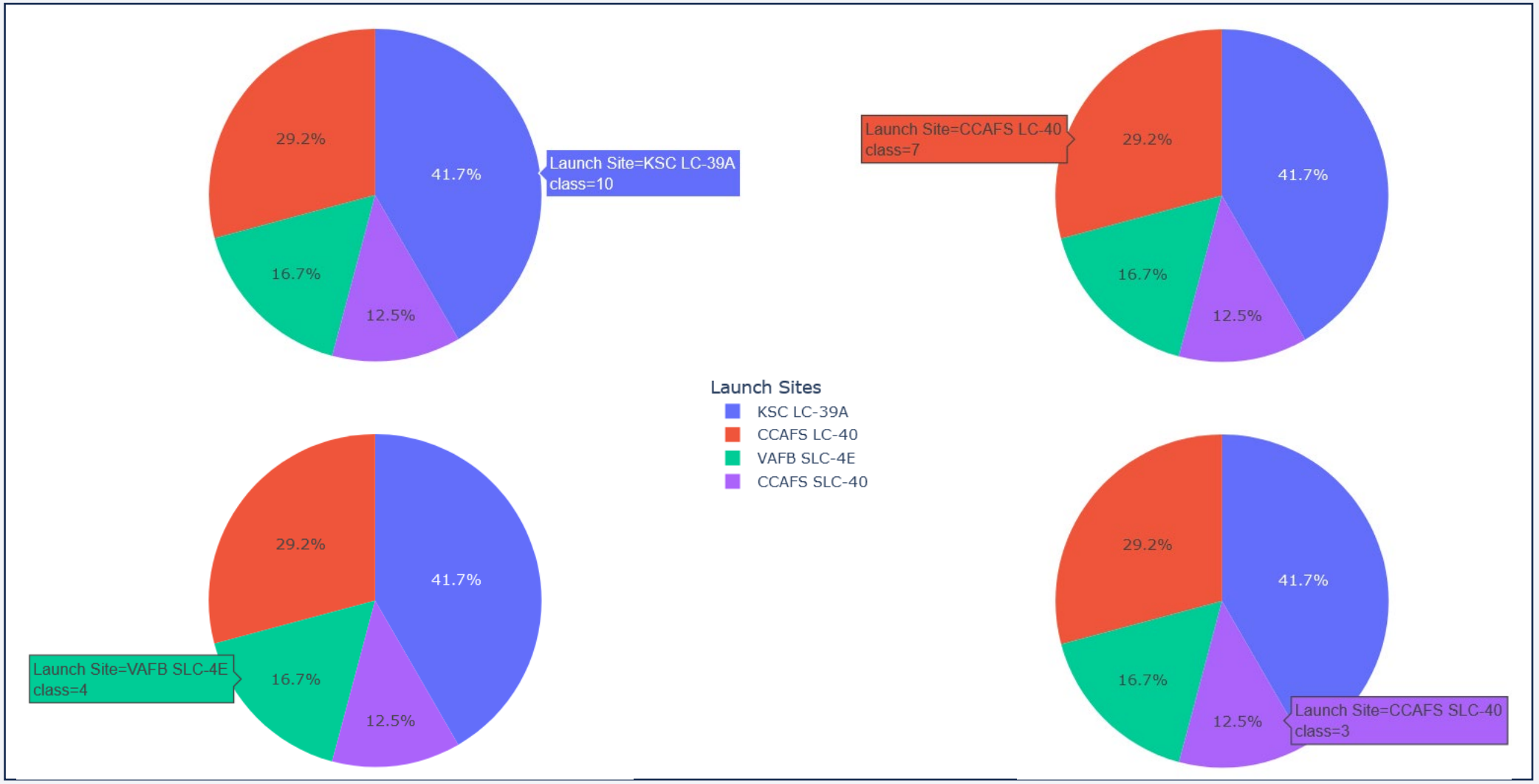



```python
### Extra ¿Is the succes ratio influenced by the height of the orbit or number of attempts?

#dictionary with attempts
attempts_dict = df['Orbit'].value_counts().to_dict()
attempts_dict

#add number of attempts
d_orbit=orbit_mean[['Orbit', 'Class']]
d_orbit['counts'] = d_orbit['Orbit'].map(attempts_dict)

#dictionary with heigth of orbits (using info from last lab and wikipedia)
km_dict = {'SO': 700, 'GTO':35786, 'ISS':408, 'MEO':2000, 'PO':600, 'LEO':2000,
           'VLEO':450, 'ES-L1':1500000,'GEO':35786,'HEO':35786,'SSO':700}


#add height
d_orbit['km'] = d_orbit['Orbit'].map(km_dict )

d_orbit[['Orbit', 'km']].sort_values(by='km')

```

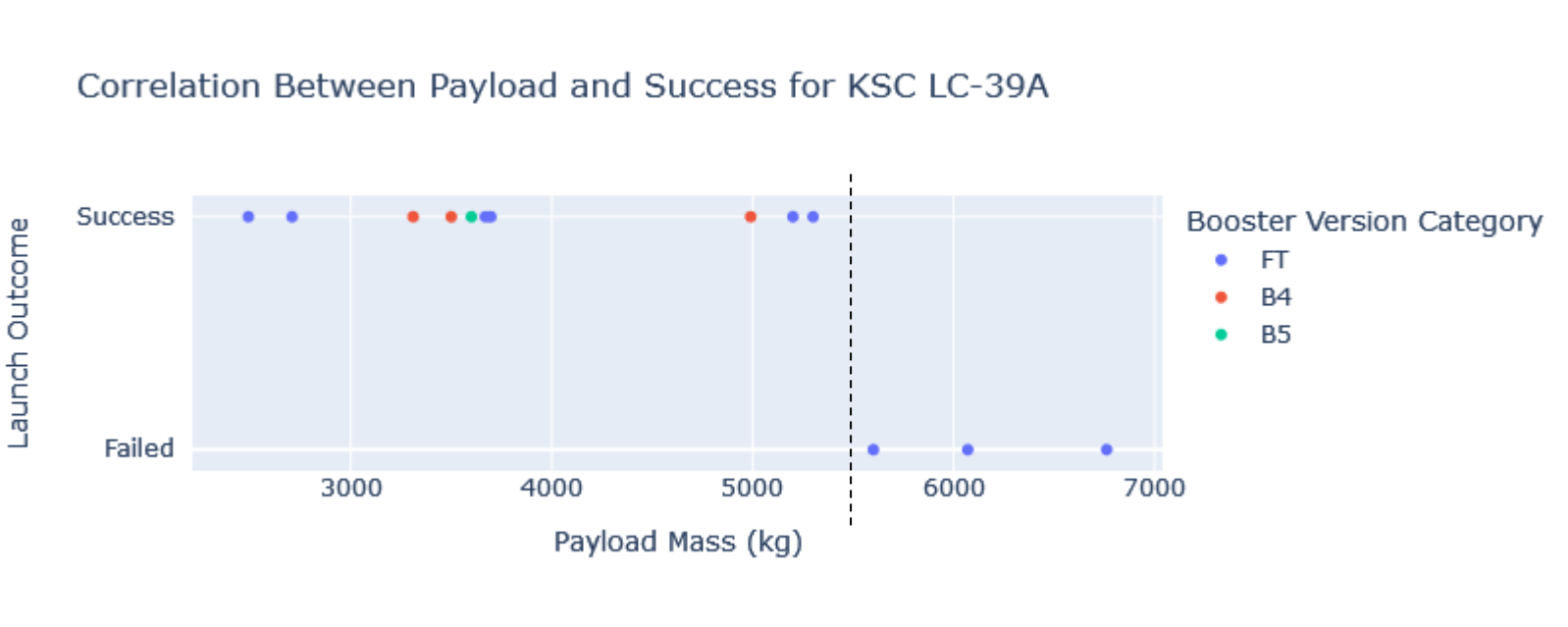
Successful outcome

- do not depend on height orbit
- do not depend on flight attempts

Appendix F: Successful Rate by Launch Site

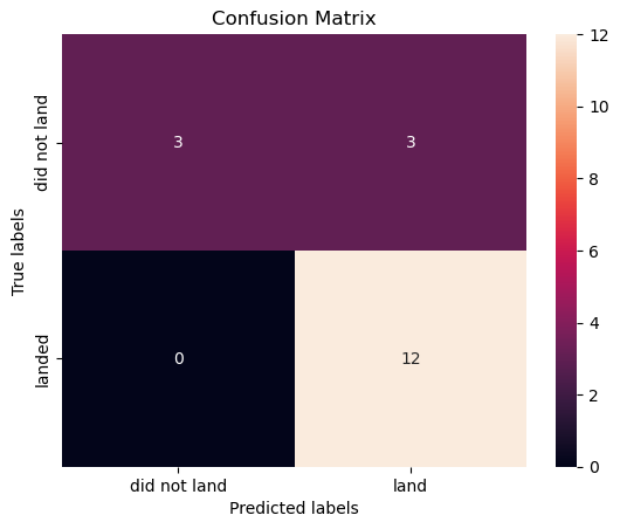


Appendix G: KSC LC-39A Payload vs Outcome



KSC LC-39A has the highest Successful Rate Launch Success Outcome is for **Payload < 5500 kg, specifically**

Appendix H: Confusion matrix models



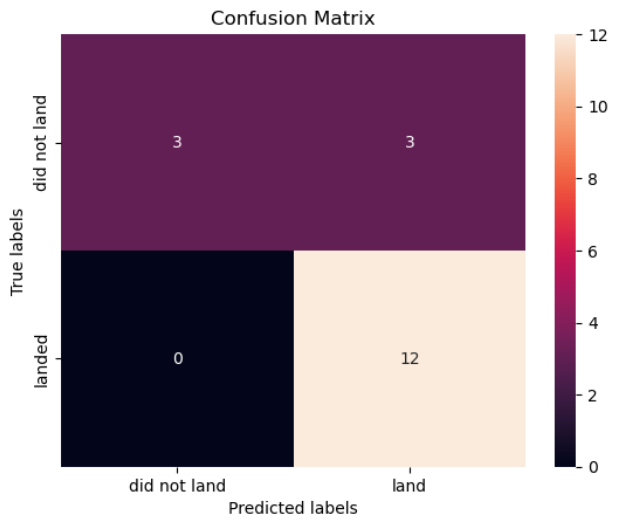
Confusion matrix Logistic

Best parameters:

'C': 0.01,

'penalty': 'l2',

'solver': 'lbfgs'



Confusion matrix Tree

Best parameters:

'criterion': 'entropy',

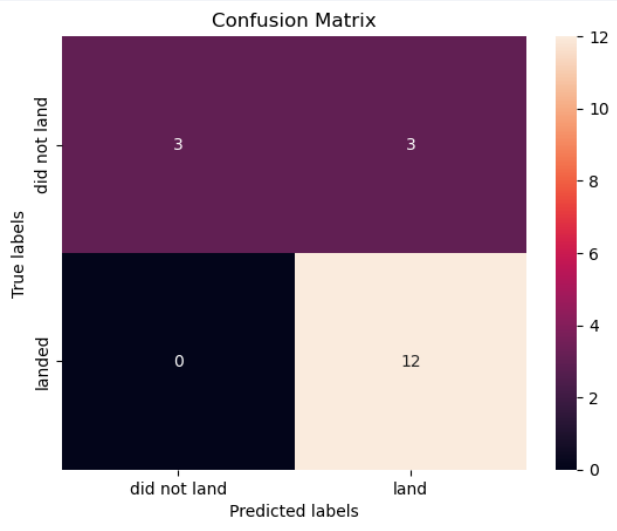
'max_depth': 4,

'max_features': 'log2',

'min_samples_leaf': 1,

'min_samples_split': 2,

'splitter': 'random'



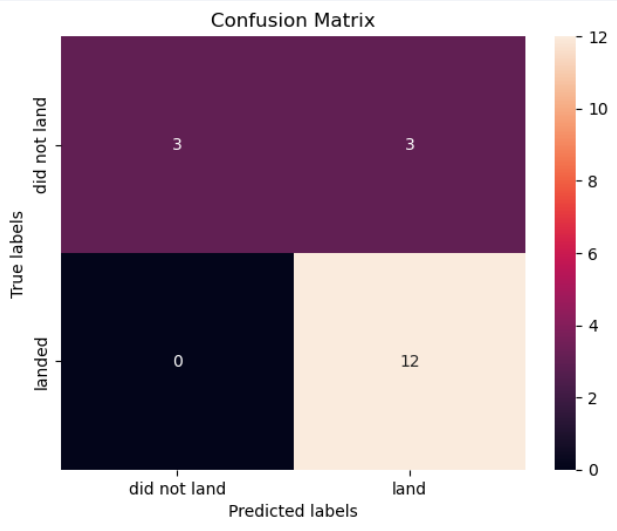
Confusion matrix SVM

Best parameters:

'C': 1.0,

'gamma': 0.0316227766,

'kernel': 'sigmoid'



Confusion matrix KNN

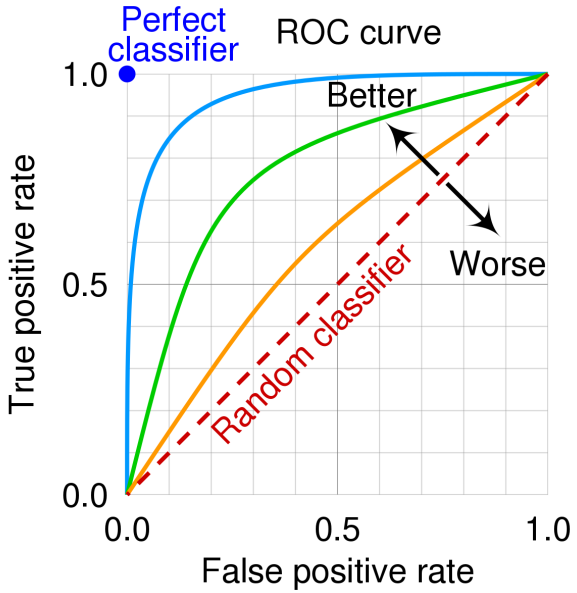
Best parameters:

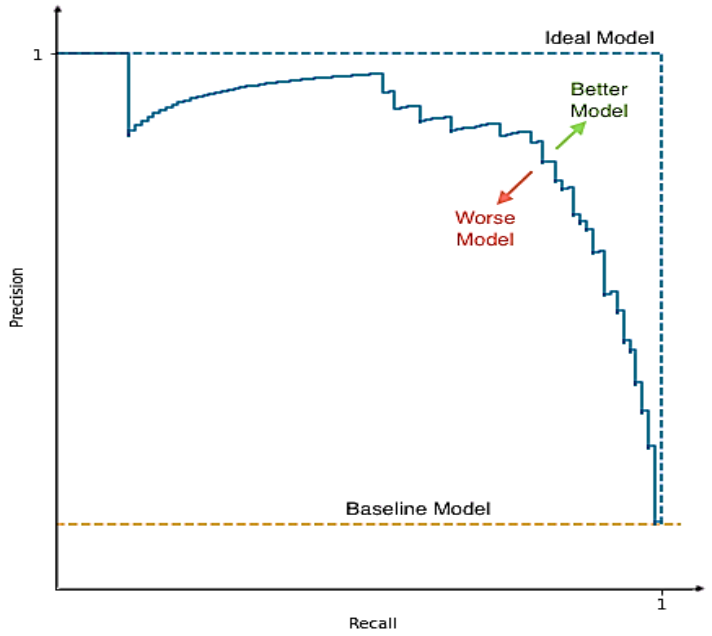
'algorithm': 'auto',
'n_neighbors': 10,

'p': 1

Appendix I: ROC and Precision-Recall curve

- Test accuracy score here, is not a good criterion to choose the better classifier.The default threshold is 0.5, and in our case, **for this threshold, we have similar scores**. Adjusting threshold allows to tune the behavior of the model for a specific problem, **in our case false positives**
- For classification model's ROC and Precision-Recall Curve are the best evaluation metrics. Both curves represents the degree or measure of separability, and they are a performance measurement for the classification problems at various threshold settings.
- They are a useful tools. The curves of different models can be compared directly in general or for different thresholds.





True Positive Rate or Recall

$$\frac{TP}{TP + FN}$$

False Positive Rate

$$\frac{FP}{FP + TN}$$

Precision

$$\frac{TP}{TP + FP}$$

References:

- https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/
- https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
- https://builtin.com/data-science/evaluating-classification-models

Thank you!