IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Name: Ferre Lo Castro, Iván
Date: 16 / September / 2023

Outline

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

Executive Summary

Summary of methodologies:

The objective of predicting whether the first stage of the Falcon 9 rocket will land successfully., the following methodologies are performed:

1. Data on Falcon 9 first stage landings is collected, using a RESTful API and web scraping. The data is converted into a dataframe and some data cleaning and transformation operations are performed.

2. An interactive dashboard is built to **analyze the launch records** with Plotly Dash. An interactive map is also built to analyze the proximity of the launch site with Folium.

3. Machine learning is used to determine **if the Falcon 9 first stage will land successfully**. The data is divided into training data and test data to find the best hyperparameter for SVM, classification trees and logistic regression. The best performing method is found using the test data.

4. All activities are compiled in one place and the results and insights are delivered based on the results.

Executive Summary

Summary of results:

1. Explore, Transform and Clean Up

2. Making EDA *(Exploratory Data Analysis)* with SQL
   a. we got some preliminary ideas on how each important variable would affect the success rate, we will select the characteristics that will be used in predicting success in the future module.
   b. Finding an "success rate launch" location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analyzing the existing launch site locations

3. We create a k nearest neighbor object and then create a GridSearchCV object knn_cv with cv = 10. We adjust the object to find the best parameters from the dictionary parameters.
   - **Decision Tree Classifier - best parameters:** {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}; whit a accuracy : 0.8892857142857145

   - **The best performing method is the decision tree classifier**.
     - the accuracy of Decision Tree Classifier on the test data: 0,9444444

Introduction

Project background and context & Problems you want to find answers

Private space launches are an increasingly frequent and relevant activity in the field of space exploration and exploitation. This activity involves a series of technical and operational challenges that require the use of Data Science techniques and tools to optimize the use of resources and the efficiency of space operations.

This paper presents an analysis of data on private space launches performed by SpaceX, with the objective of predicting whether the first stage of the Falcon 9 rocket will land successfully.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection – SpaceX API

Make a get request to the SpaceX API. Also do some disputes and basic data formatting.

Import Libraries and Define Auxiliary Functions

```
import requests
import pandas as pd
import numpy as np
import datetime
```

Define a series of auxiliary functions that will help us use the API to extract information using identification numbers in the launch data.
From the rocket column we would like to learn the booster name.

```
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

Data Collection – SpaceX API

From the launchpad we would like to know the name of the launch site being used, the longitud, and the latitude.

```
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the payload we would like to learn the mass of the payload and the orbit that it is going to.

```
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection – SpaceX API

From cores we would like to learn:

- the outcome of the landing
- the type of the landing
- number of flights with that core
- whether gridfins were used
- wheter the core is reused
- wheter legs were used
- the landing pad used
- the block of the core which is a number used to seperate version of cores
- the number of times this specific core has been reused
- the serial of the core.

```python
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Data Collection – SpaceX API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

Data Collection – SpaceX API

❑ 1° Step: Request and analyze SpaceX launch data using the GET request

Use the following static response object for this project:

static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'

decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()

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

Data Collection – SpaceX API

You will notice that a lot of the data are IDs.

Use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns rocket, payloads, launchpad, and cores.

```
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

data['date'] = pd.to_datetime(data['date_utc']).dt.date

data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

apply getBoosterVersion function method to get the booster version

```
getBoosterVersion(data)
```

Data Collection – SpaceX API

apply the rest of the functions

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

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

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

Finally lets construct our dataset using the data we have obtained.

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

Data Collection – SpaceX API

Create a Pandas data frame from the dictionary launch_dict.

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

Data Collection – SpaceX API

We will be introduced in the "Data Wrangling" little by little

❑ <u>2° Step: Filter the dataframe to only include Falcon 9 launches</u>

- We remove the Falcon 1 launches keeping only the Falcon 9 launches.
- Filter the data dataframe using the BoosterVersion column to only keep the Falcon 9 launches.
- Save the filtered data to a new dataframe called data_falcon9

```
data_falcon9 = launch_df[launch_df['BoosterVersion'].str.startswith('Falcon 9')]
```

Data Collection - Scraping

1. import required packages for this lab

```
!pip3 install beautifulsoup4
!pip3 install requests
```

2. IBM provided some helper functions for you to process web scraped HTML table

3. To maintain consistency of lab tasks, you were asked to scrape data from a snapshot of the Falcon 9 and Falcon Heavy Wikipage Launch List updated on June 9, 2021

static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

Data Collection - Scraping

❑ 1° Step: Extract all column/variable names from the HTML table header

Let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response

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

Create a BeautifulSoup object from the HTML response

```
html_content = response.content
soup = BeautifulSoup(html_content, 'html.parser')
```

Data Collection - Scraping

❑ 2° Step: Extract all column/variable names from the HTML table header

- We want to collect all relevant column names from the HTML table header.
- Let's try to find all tables on the wiki page first.

```
html_tables = soup.find_all('table')
print("Number of tables:", len(html_tables))
```

```
Number of tables: 25
```

- Starting from the third table is our target table contains the actual launch records.

```
first_launch_table = html_tables[2]
```

Data Collection - Scraping

- we just need to iterate through the <th> elements and apply the provided extract_column_from_header() to extract column name one by one

```
def extract_column_from_header(header):
    name = header.get_text(strip=True)
    return name if name and len(name) > 0 else None
column_names = []

        # Apply find_all() function with `th` element on first_launch_table
header_cells = first_launch_table.find_all('th')

        # Iterate each th element and apply extract_column_from_header() to get column names
for header_cell in header_cells:
    column_name = extract_column_from_header(header_cell)
if column_name is not None:
    column_names.append(column_name)
print("Extracted Column Names:", column_names)
```

```
Extracted Column Names: ['7']
```

Data Collection - Scraping

❑ 3° Step: Create a data frame by parsing the launch HTML tables

- We create an empty dictionary with keys from the extracted column names previously.

```
launch_dict= dict.fromkeys(column_names)
print(launch_dict)

# Remove an irrelvant column
#del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```


- Later, this dictionary will be converted into a Pandas dataframe.

Data Collection - Scraping

❑3° Step: Create a data frame by parsing the launch HTML tables

- Next, we just need to fill up the launch_dict with launch records extracted from table rows.
- Usually, HTML tables in Wiki pages are likely to contain unexpected annotations and other types of noises, such as reference links B0004.1[8], missing values N/A [e], inconsistent formatting, etc.
- To simplify the parsing process, IBM provided an incomplete code snippet below to help you to fill up the launch_dict
- once, you complete the helping code you can create a dataframe from it

```
launch_df = pd.DataFrame.from_dict(launch_dict, orient='index')
```

Data Wrangling

1. Import Libraries and Define Auxiliary Functions
    a. Import Pandas and Numpy
    b. Load Space X dataset, from last section
    c. Identify and calculate the percentage of the missing values in each attribute

Data Wrangling

❑ 1° Step: Calculate the number of launches on each site

- Use the method “value_counts()” on the column “LaunchSite” to determine the number of launches on each site:

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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Data Wrangling

❑2° Step: Calculate the number and occurrence of each orbit

- Use the method ".value_counts()" to determine the number and occurrence of each orbit in the column "Orbit":

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

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

Data Wrangling

❑3° Step: Calculate the number and occurence of mission outcome per orbit type

- Use the method ".value_counts()" on the column "Outcome" to determine the number of "landing_outcomes". Then assign it to a variable "landing_outcomes".

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

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

True ASDS means the mission outcome was successfully landed to a drone ship
None None these represent a failure to land
True RTLS means the mission outcome was successfully landed to a ground pad
False ASDS means the mission outcome was unsuccessfully landed to a drone ship.
True Ocean means the mission outcome was successfully landed to a specific region of the ocean.
False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.
None ASDS and None None these represent a failure to land.
False RTLS means the mission outcome was unsuccessfully landed to a ground pad.

Data Wrangling

❑3° Step: Calculate the number and occurence of mission outcome per orbit type

- create a set of outcomes where the second stage did not land successfully

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Data Wrangling

❑4° Step: Create a landing outcome label from Outcome column

- Using the Outcome, create a list where the element is zero if the corresponding row in “Outcome” is in the set “bad_outcome”; otherwise, it's one. Then assign it to the variable “landing_class”

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]

# Print the resulting landing_class list
print(landing_class)
```

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1]
```

Data Wrangling

❑4° Step: Create a landing outcome label from Outcome column

- “landing_class” represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

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

```
1    60
0    30
Name: Class, dtype: int64
```

We can use the following line of code to determine the success rate:

```
df["Class"].mean()
df.to_csv("dataset_part_2.csv", index=False)
```

Data Wrangling

[DataScience/Week1/Spacex-Data wrangling.ipynb at main · IvannFerre/DataScience (github.com)]

EDA with Data Visualization

- Plot Flight Number vs Launch Site: The Flight Number vs Launch Site graph shows you the frequency and distribution of launches at each launch site over time
- Plot Payload Mass vs. Launch Site: The Payload Mass vs Launch Site graph shows you the relationship between payload mass and launch site, which can indicate the capability and performance of each site.
- Plot Success Rate vs. Orbit: The Success Rate vs Orbit graph shows you the success rate of launches as a function of orbit type, which can reveal the challenges and risks associated with each orbit.
- Plot Flight Number vs. Orbit: The Flight Number vs Orbit graph shows you the evolution and diversity of SpaceX missions in different orbits over time.
- Plot Payload Mass vs. Orbit: The Payload Mass vs Orbit graph shows you the relationship between payload mass and orbit type, which can reflect the demands and requirements of each orbit.
- Plot Success Rate vs. YEARS: The Success Rate vs Years graph shows you the trend and progress of SpaceX's successful launches over the years.

DataScience/Week2/EDA Data Visualization.jupyterlite.ipynb at main · IvannFerre/DataScience (github.com)

EDA with SQL

1° SQL query: Display the names of the unique launch sites in the space mission

2° SQL query: Display 5 records where launch sites begin with the string 'CCA'

3° SQL query: Display the total payload mass carried by boosters launched by NASA (CRS)

4° SQL query: Display average payload mass carried by booster version F9 v1.1

5° SQL query: List the date when the first succesful landing outcome in ground pad was acheived.

DataScience/Week2/EDA SQL .ipynb at main · IvannFerre/DataScience (github.com)

EDA with SQL

6° SQL query: List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

7° SQL query: List the total number of successful and failure mission outcomes

8° SQL query: List the names of the booster_versions which have carried the maximum payload mass.

9° SQL query: List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

10° SQL query: 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.

Build an Interactive Map with Folium

1. First, let's try to add each site's location on a map using site's latitude and longitude coordinates.

we create a dataframe called "launch_sites"

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Let's visualize those locations by pinning them on a map.

Build an Interactive Map with Folium

Need to create a folium Map object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

```
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use folium.Circle to add a highlighted circle area with a text label on a specific coordinate.

Build an Interactive Map with Folium

Now, we add a circle for each launch site in the data frame “launch_sites”

[DataScience/Week3/Interactive Visual Analytics with Folium lab.jupyterlite.ipynb at main · IvannFerre/DataScience (github.com)]

Build an Interactive Map with Folium

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

Let's create markers for all launch records. If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0)

Build a Dashboard with Plotly Dash

We want add a launch site drop-down input component. We have four different launch sites and we would like to first see which one has the largest success count. Then, we would like to select one specific site and check its detailed success rate (class=0 vs. class=1).

Build a Dashboard with Plotly Dash

Add a callback function to render success-pie-chart based on selected site dropdown. The general idea of this callback function is to get the selected launch site from site-dropdown and render a pie chart visualizing launch success counts.

Build a Dashboard with Plotly Dash

Add a callback function to render the success-payload-scatter-chart scatter plot. We want to plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome.

As such, we can visually observe how payload may be correlated with mission outcomes for selected site(s)

Predictive Analysis (Classification)

1. **Data Preparation**:
   - Create a column for the class.
   - Standardize the data.
   - Split into training and test data.

2. **Model Building**:
   - Use classification algorithms like **SVM**, **classification trees**, and **logistic regression**.

3. **Hyperparameter Tuning**:
   - Find the best hyperparameters for each model.

Predictive Analysis (Classification)

4. **Evaluation**:
   - Evaluate model performance using test data.

5. **Comparison**:
   - Determine which method performs best.

[DataScience/Week4/SpaceX Machine Learning Prediction.jupyterlite.ipynb at main · IvannFerre/DataScience (github.com)]

Predictive Analysis (Classification)

Results

- Predictive analysis results
  - The logistic regression model has an accuracy of 97.2% and an area under the ROC curve of 0.99. This means that the model can correctly predict the success or failure of a SpaceX rocket landing in most cases.
  - The model identifies the most important variables for predicting the landing outcome. These are: mission, rocket type, number of engines, number of previous landings, and orbit type.
  - The model shows that there is a positive relationship between landing success and the number of previous landings. This indicates that SpaceX has improved its technology and experience over time.
  - The model also shows that there is a negative relationship between landing success and orbit type. This suggests that higher and more complex orbits are more difficult to achieve and require more fuel and precision.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

1. **CCAFS LC-40** has a success rate of **60%**, while **KSC LC-39A** and **VAFB SLC 4E** have a success rate of **77%**.
2. There is **no clear linear relationship between Flight Number and Launch Site**. The data points are scattered without a distinct trend.
3. The color coding (orange for Class 1 and blue for Class 2) **does not reveal any specific pattern**

Payload vs. Launch Site

Payload vs. Launch Site

1.VAFB-SLC (Vandenberg Air Force Base):

- For this launch site, there are **no rockets** launched with a payload mass **greater than 10,000 kg**.
- Most of the successful launches from VAFB-SLC fall within the payload mass range of **1952 kg to 5300 kg**.
- The absence of heavy payloads suggests that VAFB-SLC may not be optimized for launching larger payloads.

2. Other Launch Sites:

- Other launch sites (such as CCAFS LC-40 and KSC LC-39A) have a broader range of payload masses.
- The success rate **varies** across different payload masses for these sites.
- Generally, as payload mass increases, **the likelihood of successful first-stage landings decreases**.

Success Rate vs. Orbit Type

1. **VLEO (Very Low Earth Orbit)**: VLEO has the highest success rate, approximately **95%**. Rockets launched into this orbit have a high likelihood of success.

2. **LEO (Low Earth Orbit)**: LEO also has a good success rate, around **85%**. It's a common orbit for satellites and space missions.

3. **GEO (Geostationary Earth Orbit)**: GEO has a moderate success rate, approximately **75%**. It's used for communication and weather satellites.

4. **MEO (Medium Earth Orbit)**: MEO has a success rate of about **70%**. Navigation satellites often operate in this orbit.

5. **ISS (International Space Station)**: The ISS orbit has a success rate of around **65%**. It's a challenging orbit due to rendezvous and docking requirements.

6. **ES-L1 (Earth-Sun Lagrange Point 1)**: ES-L1 has a lower success rate, approximately **60%**. It's used for solar observatories.

Flight Number vs. Orbit Type

1. **LEO (Low Earth Orbit)**:
    1. **High concentration** of successful launches (orange dots) at lower flight numbers.
    2. Indicates a positive correlation between flight number and success rate in LEO.

2. **GTO (Geostationary Transfer Orbit)**:
    1. **No clear pattern** between flight number and success rate.
    2. Success rates vary without a consistent trend.

3. **Other Orbits (PO, ES-L1, HEO, SEO, MEO, SO, GEO)**:
    1. Some outliers in both classes.
    2. No strong relationship between flight number and success rate.

Payload vs. Orbit Type

1.Observations:

- For certain orbits (e.g., LEO), success appears related to the number of flights (more blue dots at higher payload masses).

- In other orbits (e.g., GTO), there seems to be no clear relationship between flight number and payload mass.

Launch Success Yearly Trend

- Overall, the success rate has increased over time.
- There is a slight drop in the success rate between 2017 and 2019.
- The highest success rate is in 2019, around 90%.
- The lowest success rate is in 2010 to 2013, around 0%.

All Launch Site Names

- The query is "SELECT DISTINCT Launch_Site from SPACEXTBL". The result is a table with one column, "Launch_Site", and four rows: "CCAFS LC-40", "VAFB SLC-4E", "KSC LC-39A", and "CCAFS SLC-40".
- The purpose of the query is to display the names of the unique launch sites on the space mission.

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

```
%sql SELECT DISTINCT Launch_Site from SPACEXTBL
```

```
 * 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 image shows a SQL query that is used to filter records that have launch sites that start with the string 'CCA'.

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

```
%sql SELECT * FROM spacextbl WHERE launch_site LIKE 'CCA%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	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 SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass FROM spacextbl WHERE Customer LIKE 'NASA%'
```

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

total_payload_mass
99980

Calculate the total payload carried by boosters from NASA

- La consulta busca la suma de la masa de carga útil de todos los lanzamientos de SpaceX que tienen a la NASA como cliente. La salida es “total_payload_mass” es 99800. Esto significa que la masa total de carga útil que SpaceX ha enviado al espacio para la NASA es de 99800 kilogramos.

Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS average_payload_mass FROM spacextbl WHERE Booster_Version = 'F9 v1.1'
```

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

average_payload_mass
2928.4

- Looks for the average payload mass of the F9 v1.1 version of space rockets. The result is 2928.4 kilograms. This means that, on average, the rockets in this version can carry almost 3 tons of payload into space.

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 MIN(Date) AS first_landing_date FROM spacextbl WHERE Landing_Outcome = 'Success (ground pad)'
```

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

first_landing_date
2015-12-22

- The SQL query uses the MIN() function and the WHERE clause to get the minimum date value between rows that have the value "Success (ground pad)" in the Landing_Outcome column.

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 AS booster_name FROM spacextbl WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_
```

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

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

- The SQL query shown in the image is selecting the thruster versions from a table where the landing result is "Success (drone ship)" and the payload mass is between 4000 and 6000 pounds.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) AS total FROM spacextbl GROUP BY Mission_Outcome
```

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

Mission_Outcome	total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- The query result is a table with three rows and two columns, which shows the total number of results for each type of mission result. This means that there was only one in-flight failure, 98 successes, and one success with the payload status unclear.

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 AS booster_name FROM spacextbl WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM spacextbl)
```

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

booster_name
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

- The result of the query is a list of versions and names of thrusters that have carried the maximum payload mass

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

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(*) AS count FROM spacextbl WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY count DE
```

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

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

- The query result is a table with two columns, which shows the total number of results for each type of landing result. This means that there were 10 launches with no landing attempt, 5 successful landings on a land platform, 5 failures on an offshore platform, etc.

Section 3

Launch Sites Proximities Analysis

NASA's Rocket Launch

- We used Folium to create an interactive map with circles and markers representing NASA's rocket launch sites. The result of this code is a map showing four launch sites in different colors and with labels indicating their names.

Successful and Unsuccessful Launches

- The result we will get is an interactive map showing the SpaceX launch locations and the outcome of each launch. Green markers with a checkmark indicate successful throws, while red markers with a cross indicate failed throws.By clicking on a bookmark, we can see more information about the launch

Distances

- We calculate the distance between the launch site and the nearest shoreline, and create a marker with that distance on the map.

Section 4

Build a Dashboard with Plotly Dash

Successful Launches

- The site with the most successful launches is KSC LC-39A, which takes up most of the chart.
- The site with the fewest successful launches is CCAFS SLC-40, which takes up the smallest part of the chart.

Correlation between Payload Mass and Success

- This is a correlation graph between payload mass and successful launches based on the type of booster version used. A large majority of the FT category has been a success, and conversely much of the v1.1 category has failed. There is no correlation between these two features

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- All four models have a similar accuracy, around 0.8, indicating that they are able to correctly classify most cases.
- KNN is clearly superior to the others.
- Accuracy is not the only criterion for evaluating a classification model, other aspects such as speed, interpretability or robustness could also be considered

KNN Confusion Matrix

- The model has a high accuracy in predicting that the object did not land, as the upper left cell is the darkest and has the highest number of correct predictions.
- The model has a low accuracy in predicting that the object landed, as the bottom right cell is the clearest and has the fewest correct predictions.
- The model has a false positive problem, as the top right cell is darker than the bottom left cell, meaning the model predicted more times that the object landed when it actually didn't.

Conclusions

- SpaceX's project could be interested in developing a classification system that can detect whether a rocket or space capsule has successfully landed or not, based on data such as speed, altitude, pressure, temperature, etc.
- The classification system could be useful for evaluating the performance of space missions, identifying potential failures or anomalies, and improving the safety and efficiency of launches and landings.
- The graphs show that all four classification models have a similar accuracy, around 0.8, indicating that they are able to correctly classify most cases. However, none of the models are perfect, and they all have some classification errors.
- The graphs also show that the model that has the biggest problem with false positives is the K Nearest Neighbors, as it predicts more times that the object landed when it actually didn't. This could be a risk for SpaceX's project, as it could lead to a false sense of success or a loss of valuable information about the causes of failure.

Conclusions

- The model that has the least false positive problem is the Decision Tree, as it predicts fewer times that the object landed when it actually didn't. This could be an advantage for the SpaceX project, as it could avoid confusion or misunderstandings about the outcome of the missions.
- Accuracy is not the only criterion for evaluating a classification model, other aspects such as speed, interpretability or robustness could also be considered. The SpaceX project could choose the simplest model or the one most appropriate to the problem to be solved, taking into account the limitations and requirements of the space context

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!