Data Science and Machine Learning Capstone Project

Begin your course today Start course

Expand all

Welcome

- Course Introduction 1 min
- General Information 1 min
- Syllabus 1 min
- Grading Scheme 1 min

Module 1 - Capstone Introduction and Understanding the Datasets

- Module Introduction and Learning Objectives 1 min
- (Video) Project Scenario and Overview (3:08) 4 min
- Hands-on Lab: IBM Cloud Account Creation and Watson Studio 1 min + 1 activity
- Hands-on Lab: Getting Started with GitHub 1 min
- Hands-on Lab: Publishing Notebook to GitHub 1 min
- Hands-on Lab: IBM Studio Setup and Project Creation 1 min
- Hands-on Lab: Adding a Notebook to the Project 1 min
- (Video) Data Collection Overview (4:15) 5 min
- Hands-on Lab: Complete the Data Collection API Lab 1 min
- Hands-on Lab: Complete the Data Collection with Web Scraping 1 min
- Hands-on Lab: Rest API 1 min + 1 activity
- Check Points: Rest API 1 activity
- Graded Quiz: Web Scraping (4 Questions) 1 activity
  Quiz due Mar 26, 2022, 3:06 PM GMT+8
- (Video) Data Wrangling Overview (2:12) 3 min
- Hands-on Lab: Data Wrangling 1 min
- Check Points: Data Wrangling 1 activity
- Graded Quiz: Data Wrangling (4 Questions) 1 activity
  Quiz due Mar 26, 2022, 3:06 PM GMT+8

Module 2 - Exploratory Data Analysis (EDA)

- Module Introduction and Learning Objectives 1 min
- (Video) Exploratory Data Analysis Overview (2:01) 3 min
- Hands-on Lab: Complete the EDA with SQL 1 min + 1 activity
- Check Points: Exploratory Analysis using SQL 1 activity
- Graded Quiz: Exploratory Analysis using SQL (5 Questions) 1 activity
  Quiz due Mar 31, 2022, 7:06 AM GMT+8
- Hands-on Lab: Complete the EDA with Visualization(70 mins) 1 min
- Check Points: Complete the EDA with Visualization 1 activity
- Graded Quiz: Complete the EDA with Visualization (3 Questions) 1 activity
  Quiz due Mar 31, 2022, 7:06 AM GMT+8

Module 3 - Interactive Visual Analytics and Dashboard

- Module Introduction and Learning Objectives 1 min
- (Video) Data Visualization and Dashboard Overview (1:54) 2 min
- Hands-on Lab: Interactive Visual Analytics with Folium 1 min
- Hands-on Lab: Build an Interactive Dashboard with Ploty Dash 1 min + 1 activity
- Check Points: Interactive Visual Analytics and Dashboard 1 activity
- Graded Quiz: Interactive Visual Analytics and Dashboard (5 Questions) 1 activity
  Quiz due Apr 4, 2022, 11:06 PM GMT+8

Module 4 - Predictive Analysis (Classification)

- Module Introduction and Learning Objectives 1 min
- (Video) Predictive Analysis Overview (1:01) 2 min
- Hands on Lab: Complete the Machine Learning Prediction lab 1 min
- Check Points: Predictive Analysis 1 activity
- Graded Quiz: Predictive Analysis (3 Questions) 1 activity
  Quiz due Apr 9, 2022, 3:06 PM GMT+8

Module 5 - Present Your Data-Driven Insights

- (Video) Elements Of A Successful Data Findings Report (4:49) 5 min
- (Reading) Structure of a Report 1 min
- (Video) Best Practices For Presenting Your Findings (3:10) 4 min
- (Optional) Hands-on Lab: Getting Started With PowerPoint For The Web 1 min
- (Optional) Hands-on Lab: Basics of PowerPoint 1 min
- (Optional) Hands-on Lab: Save your PowerPoint Presentation as PDF 1 min
- Final Presentation - Submission Overview and Instructions 1 min
- Exercise: Preparing Your Presentation (with provided slide template) 1 min
- Peer Review: Submit your Work and Review your Peers (1 Question) 1 min + 1 activity

Welcome !

Hello! And welcome to this capstone course.

Congratulations for making it this far! My name is Joseph Santarcangelo and Yan Luo. We are pleased to be your instructors for this capstone course. You will apply your data science skills as a Data scientist for a private space launch company in this project.

As a starting point of almost all data science projects, you need to collect data, as much and relevant as possible.

You will be collecting data from various sources. After your raw data has been collected, you will need to improve the quality by performing data wrangling.

Then you can start exploring the processed data. We will be your guide as we explore some really interesting real-world datasets together. You'll get to practice your SQL skills as we query the data and gather insights.

You'll gain further insights into the data by applying some basic statistical analysis and data visualization, you'll be able to see directly how variables might be related to each other.

We'll drill down into finer levels of detail by splitting the data into groups defined by categorical variables or factors in your data.

You will be guided to build, evaluate, and refine predictive models for discovering more exciting insights.

The final task of this capstone project is to create a presentation that will be developed into stories of all your analysis.

Thanks and good luck!

About the Course

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Prerequisites

To successfully complete this capstone project, you must understand how to use data science techniques such as data analysis, data visualization, and machine learning by using Python.

Before taking this course, you should have already completed the following courses:

- Python Basics for Data Science
- Analyzing Data with Python
- Visualizing Data with Python
- Machine Learning with Python: A Practical Introduction

Syllabus

Module 1 - Capstone Introduction and Understanding the Datasets

- Module Introduction & Objectives
- Course Introduction
- Project Scenario and Overview
- IBM Watson Account creation and Watson Studio
- Import Notebook to Watson Studio
- Getting started with Github
- Publishing Notebook to Github
- Data Collection Overview
- Web scraping related datasets
- REST API calls
- Check points (ungraded)
- Graded quiz
- Data wrangling overview
- Data wrangling
- Check points (ungraded)
- Graded quiz

Module 2 - Wrangling the data

- Module Introduction & Objectives
- Exploratory data analysis overview
- Exploratory Analysis using SQL (and load dataset on IBM Db2)
- Check points (ungraded)
- Graded quiz
- Exploratory data analysis using Pandas and Matplotlib
- Check points (ungraded)
- Graded quiz

Module 3 - Interactive visual analytics and Dashboard

- Module Introduction & Objectives
- Data visualization and dashboard overview
- Launch Site Location Analysis with Folium
- Build an interactive dashboard with Ploty Dash
- Check points (ungraded)
- Graded quiz

Module 4 - Predictive Analysis (Classification)

- Module Introduction & Objectives
- Predictive analysis overview
- Build and evaluate classification models
- Check points (ungraded)
- Graded quiz

Module 5 - Present Your Data-Driven Insights

- Module Introduction & Objectives
- Elements Of A Successful Data Findings Report
- Structure of a Report
- Best Practices For Presenting Your Findings
- Optional: Getting Started with PowerPoint for the Web
- Optional: Basics of PowerPoint
- Optional: Save your PowerPoint Presentation as PDF
- Final submission overview and instructions
- Exercise: preparing your presentation (provided a slide template)
- Peer-review rubric and prompts based on completed slide

Grading Scheme

Bookmarked

GRADING SCHEME

This section contains information for those earning a certificate. Those auditing the course can skip this section and click next.

1. The course contains 6 Graded Quizzes. Each Graded Quiz carries an equal weight of 10% of the total grade.
2. The minimum passing mark for the **course** is 70%.
3. Permitted attempts are per **question**:
   - One attempt - For True/False questions
   - Two attempts - For any question other than True/False
4. There are no penalties for incorrect attempts.
5. Clicking the "**Submit**" button when it appears, means your submission is **FINAL**. You will **NOT** be able to resubmit your answer to that question again.
6. Check your grades in the course at any time by clicking on the "Progress" tab.

Module Introduction and Learning Objectives

Module Introduction

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.

Learning Objectives

- Use your data analysis tools to load a dataset, clean it, and find out interesting insights from it.
- Use data science methodologies to define and formulate a real-world business problem.

Project Scenario and Overview

The commercial space age is here, companies are making space travel affordable for everyone. Virgin Galactic is providing suborbital spaceflights. Rocket Lab is a small satellite provider. Blue Origin manufactures sub-orbital and orbital reusable rockets.

Perhaps the most successful is **SpaceX.** SpaceX's accomplishments include: Sending spacecraft to the International Space Station. Starlink, a satellite internet constellation providing satellite Internet access. Sending manned missions to Space.

One reason SpaceX can do this is the rocket launches are relatively inexpensive. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch.

Spaces X's Falcon 9 launch like regular rockets. To help us understand the scale of the Falcon 9, we are going to use these diagrams from Forest Katsch, at zlsadesign.com. He is a 3D artist and software engineer. He makes infographics on spaceflight and spacecraft art. He also makes software. The payload is enclosed in the fairings. Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage.

The first stage is shown here. This stage does most of the work and is much larger than the second stage. Here we see the first stage next to a person and several other landmarks. This stage is quite large and expensive. Unlike other rocket providers, SpaceX's Falcon 9 Can recover the first stage. Sometimes the first stage does not land. Sometimes it will crash as shown in this clip. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer.

In this capstone, you will take the role of a data scientist working for a new rocket company. Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. Your job is to determine the price of each launch. You will do this by gathering information about Space X and creating dashboards for your team. You will also determine if SpaceX will reuse the first stage.

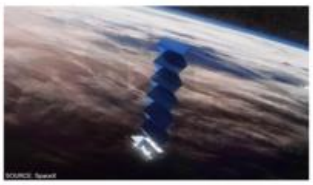
Instead of using rocket science to determine if the first stage will land successfully, you will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Project Scenario and Overview

Galactic, Rocket Lab and BLUE ORIGIN

SPACEX

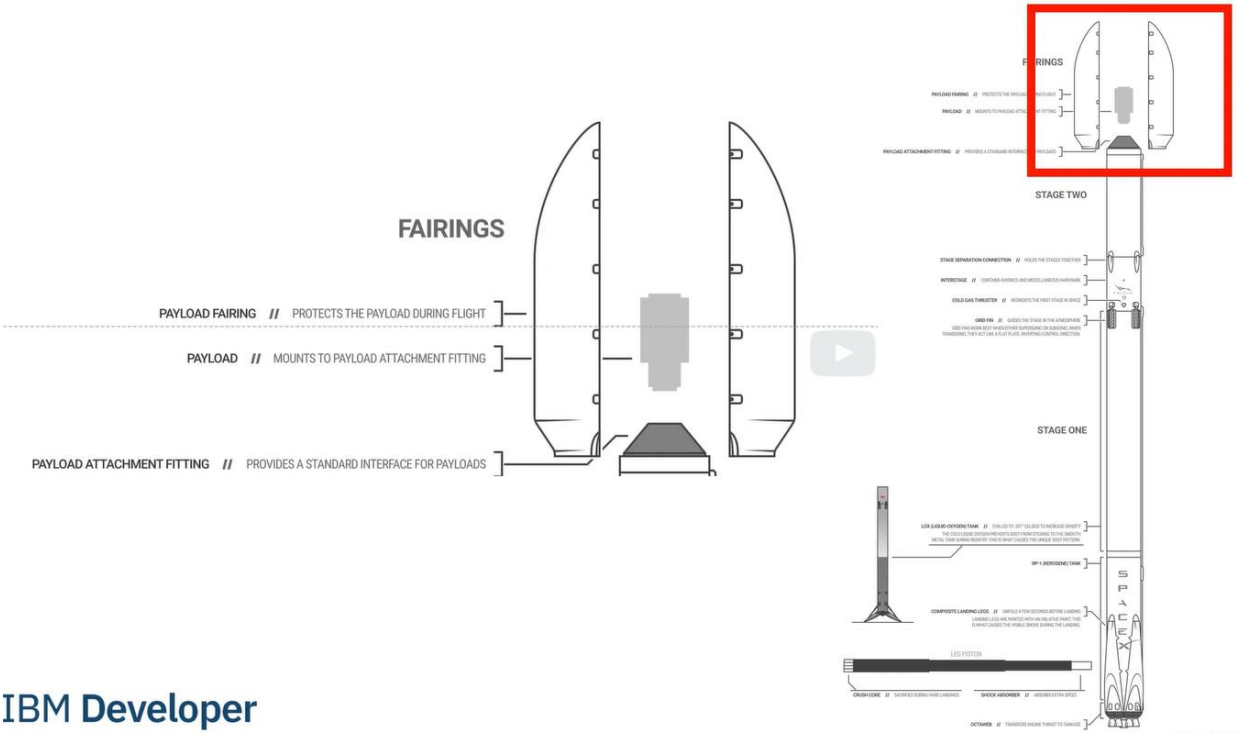
www.flickr.com



Source Space X

Source NASA

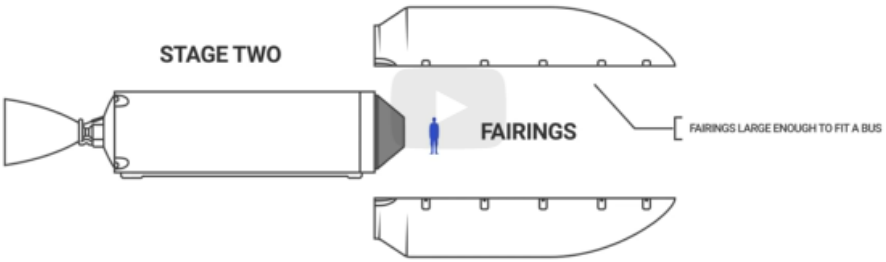
SpaceX's accomplishments include: Sending spacecraft to the International Space Station. Starlink, a satellite internet constellation providing satellite Internet access. Sending manned missions to Space.



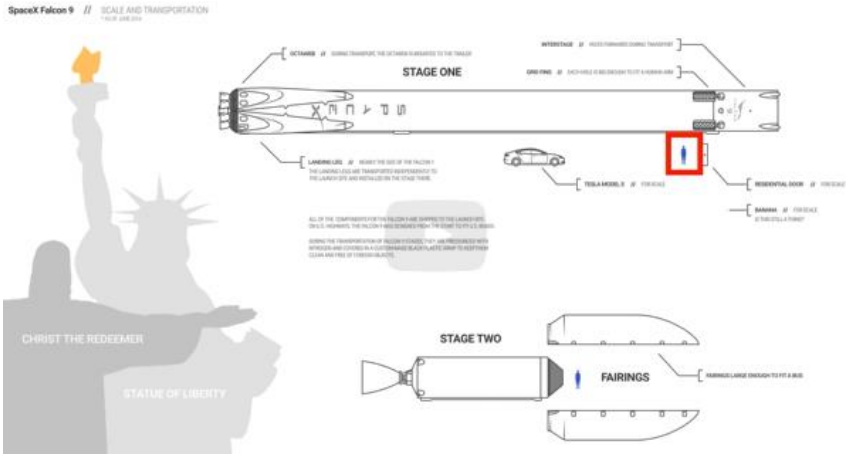
To help us understand the scale of the Falcon 9, we are going to use these diagrams from Forest Katsch,

IBM Developer

at zlsadesign.com. He is a 3D artist and software engineer. He makes infographics on spaceflight and spacecraft art. He also makes software. The payload is enclosed in the fairings.



Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage



The first stage is shown here. This stage does most of the work and is much larger than the second stage. Here we see the first stage next to a person and several other landmarks. This stage is quite large and expensive

SPACE Y

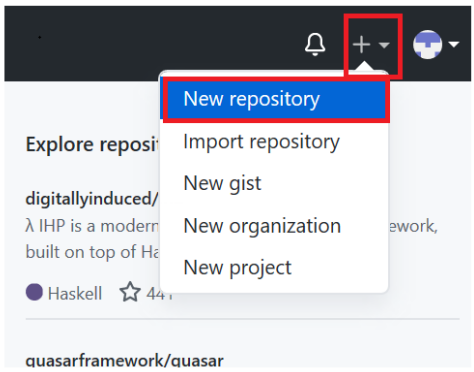
Allon Mask

Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Musk. Your job is to determine the price of each launch. You will do this by gathering information about Space X and creating dashboards for your team. You will also determine if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, you will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

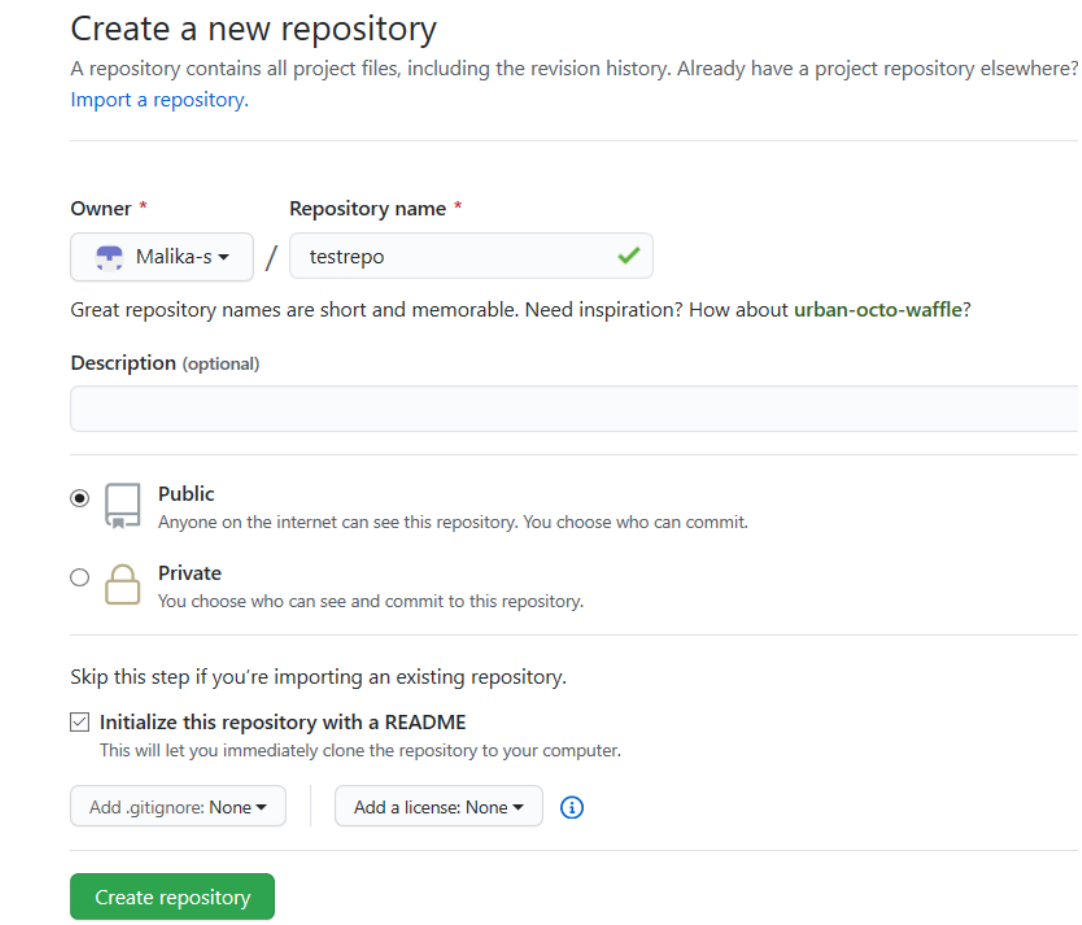
On **Github:** To Create a Repository

Exercise 2: Adding a Project / Repo

Step 1: Click on the + symbol and click `New repository`.



Step 2: Provide a repository a name and initialize with the empty `README.md` file.



and click `Create repository`.

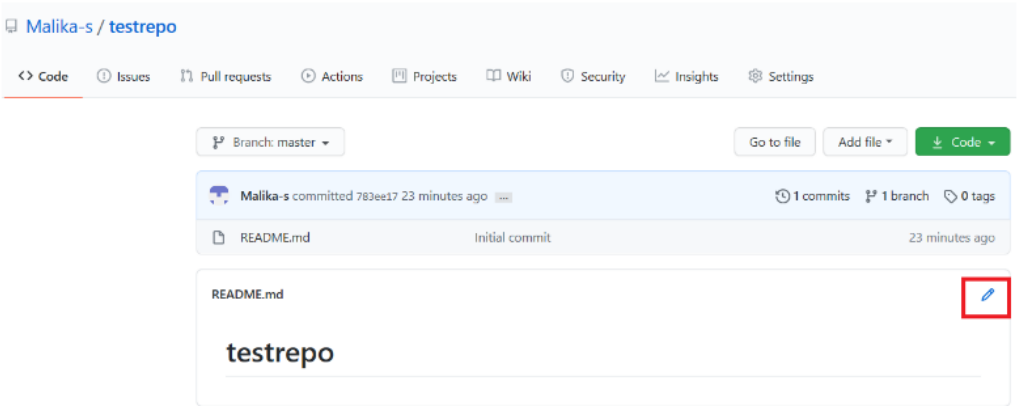
Now, you will be redirected to the repository you have created.

To Create/Edit A File

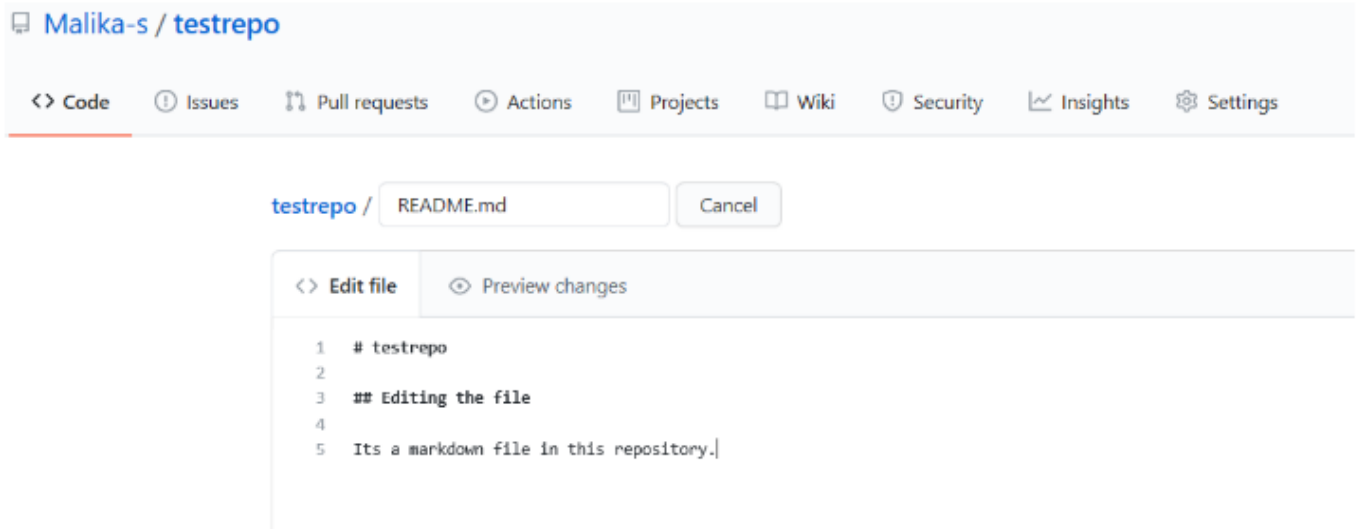
Exercise 3: Create / edit a file

Exercise 3a: Edit a file

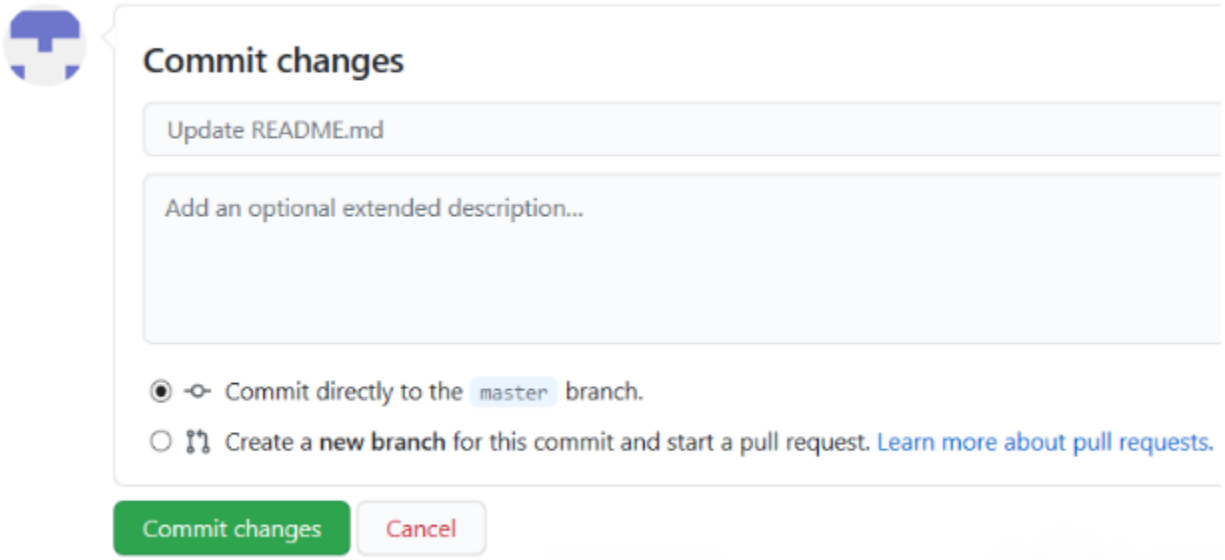
Step 1: Once the repository is created, the root folder of your repository is listed by default and it has just one file `ReadMe.md`. Click on the pencil icon to edit the file



Step 2: Add text to file.



Step 3: Scroll down the page after adding the text and click `Commit Changes`.

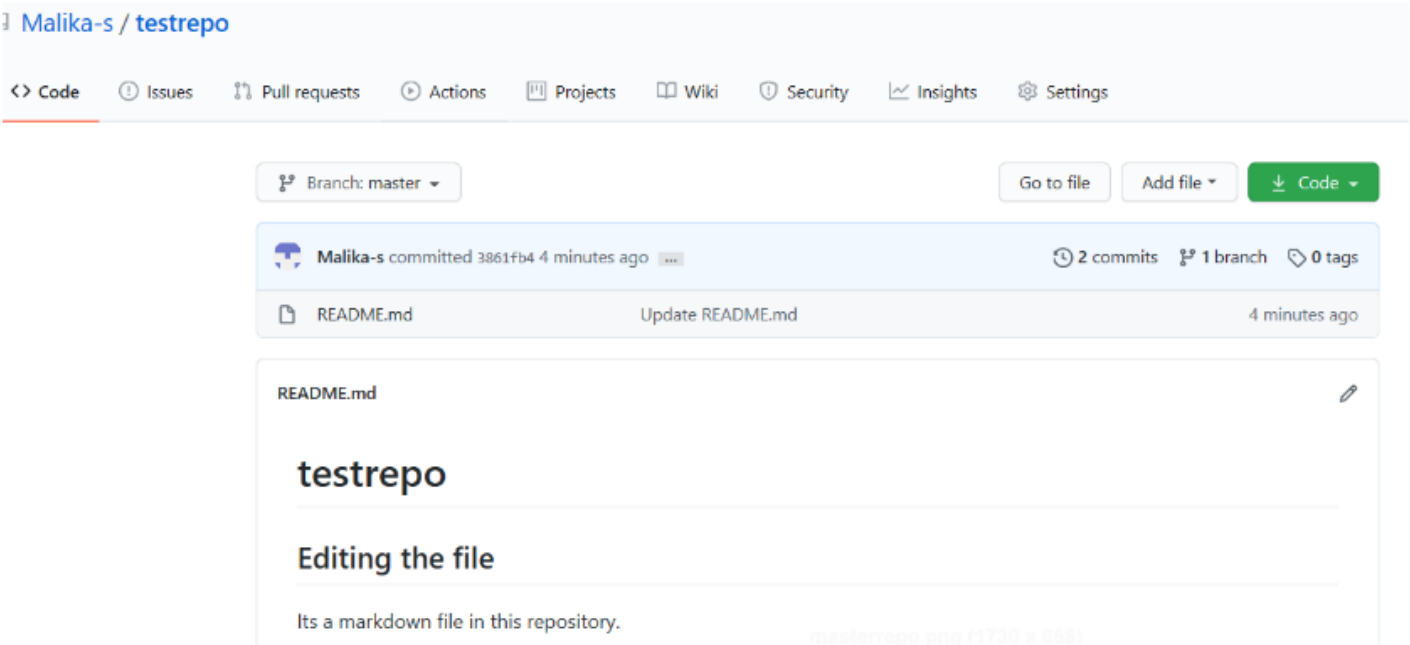


Now, check your file is edited with the new text.

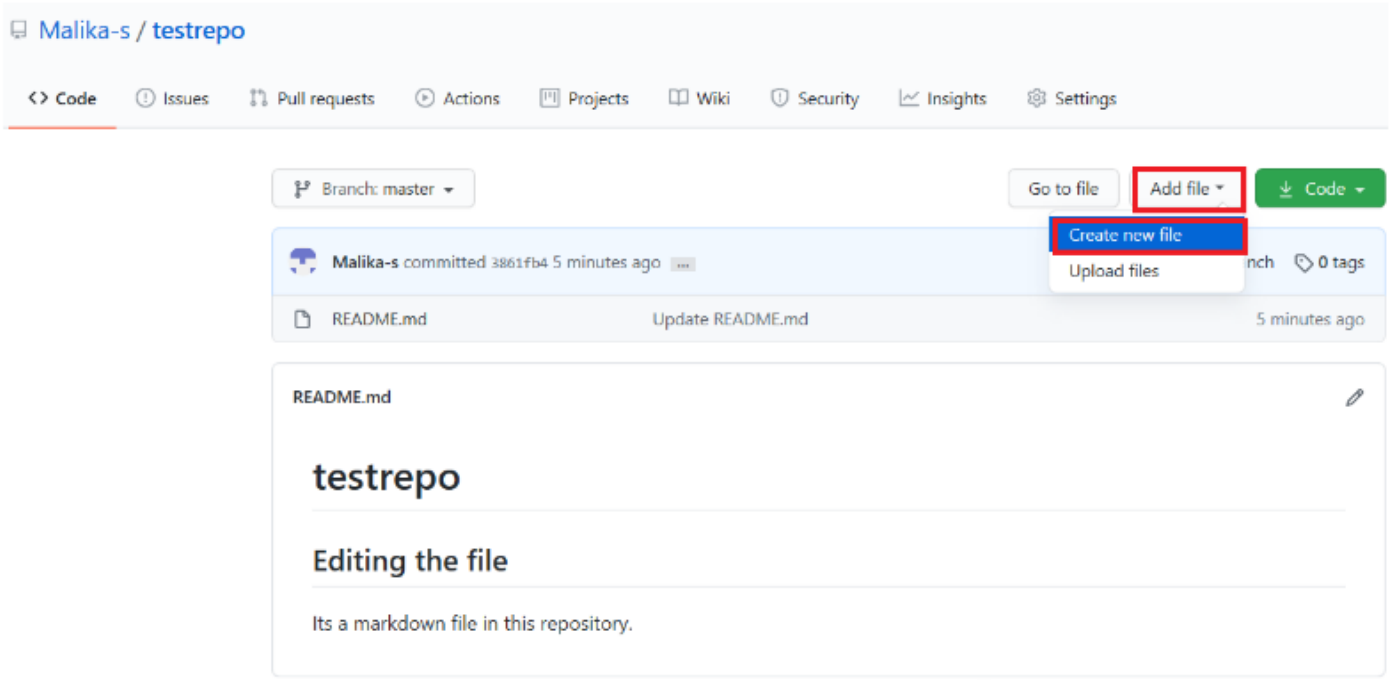
To Create A New File

Exercise 3b: Create a new file

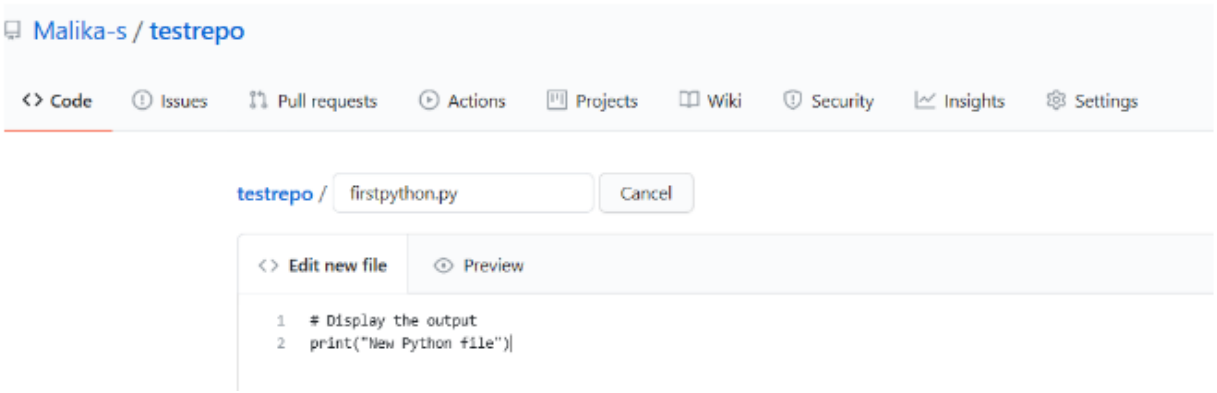
Step 1: Click on the repository name to go back to the master branch like in this testrepo.



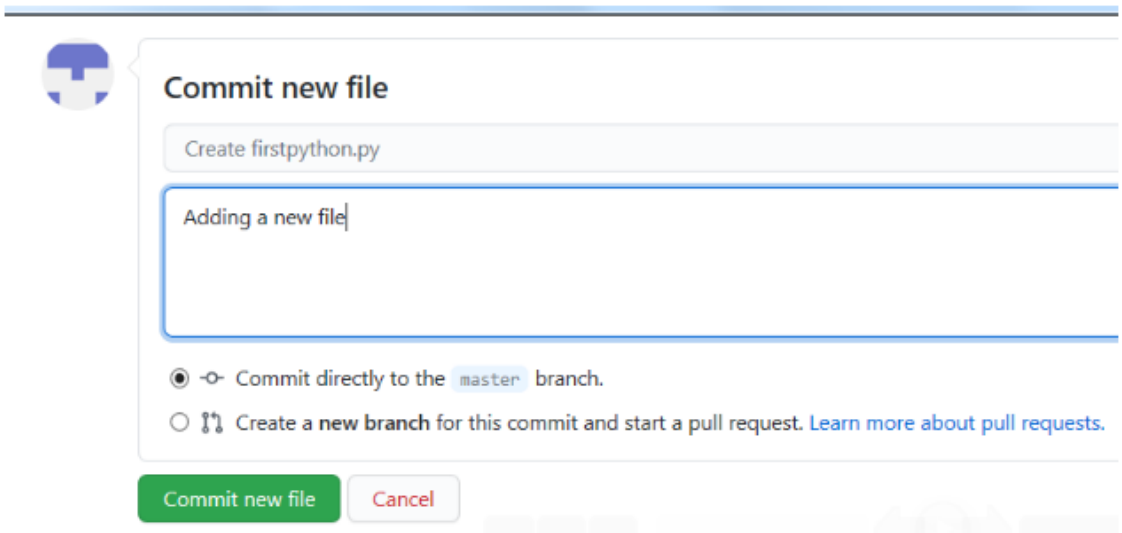
Step 2: Click `Add file` and select `Create New file` to create a file in the repository.



Step 3: Provide the file name and the extension of the file. For example, firstpython.py and add the lines.



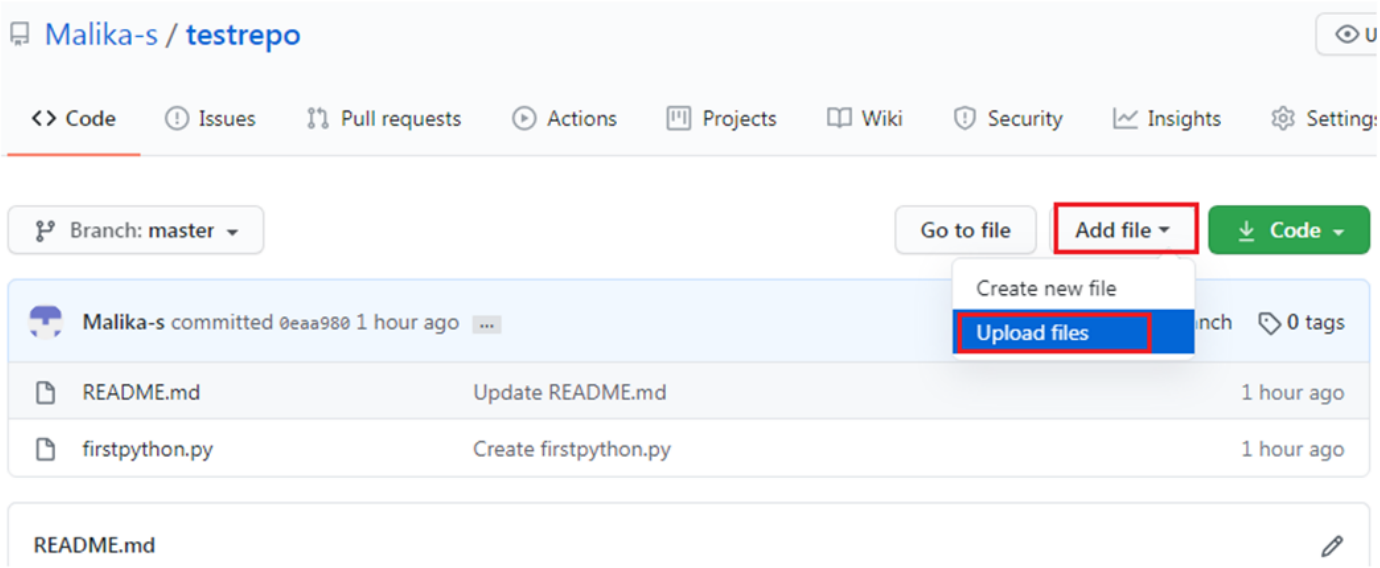
Step 4: Scroll down the page after adding the text. `Add description` of the file (optional) and click `Commit` new file.



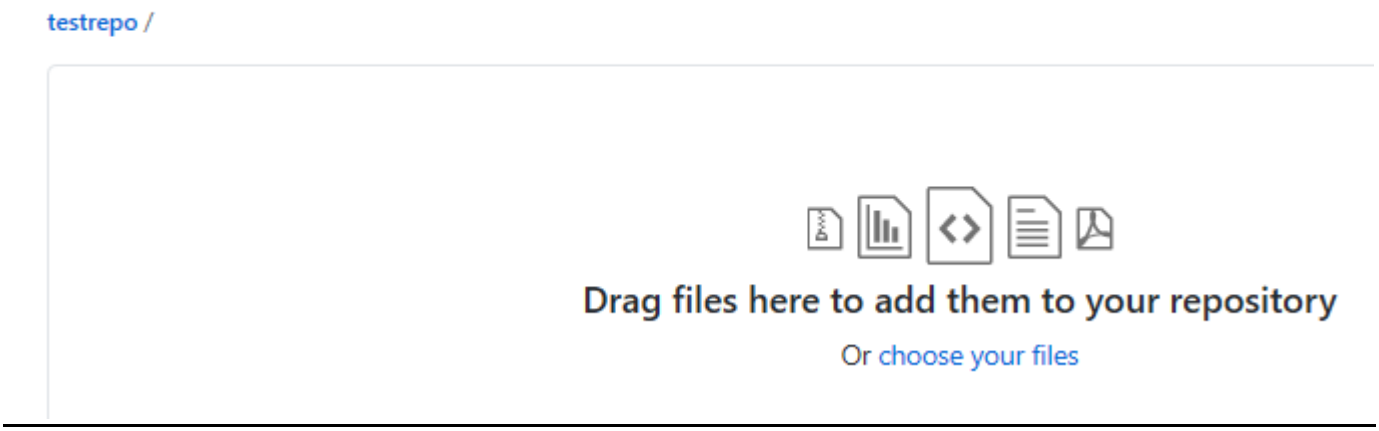
Step 5: Your file is now added to your repository and the repository listing shows when the file was added/changed.

To Upload a File and Commit

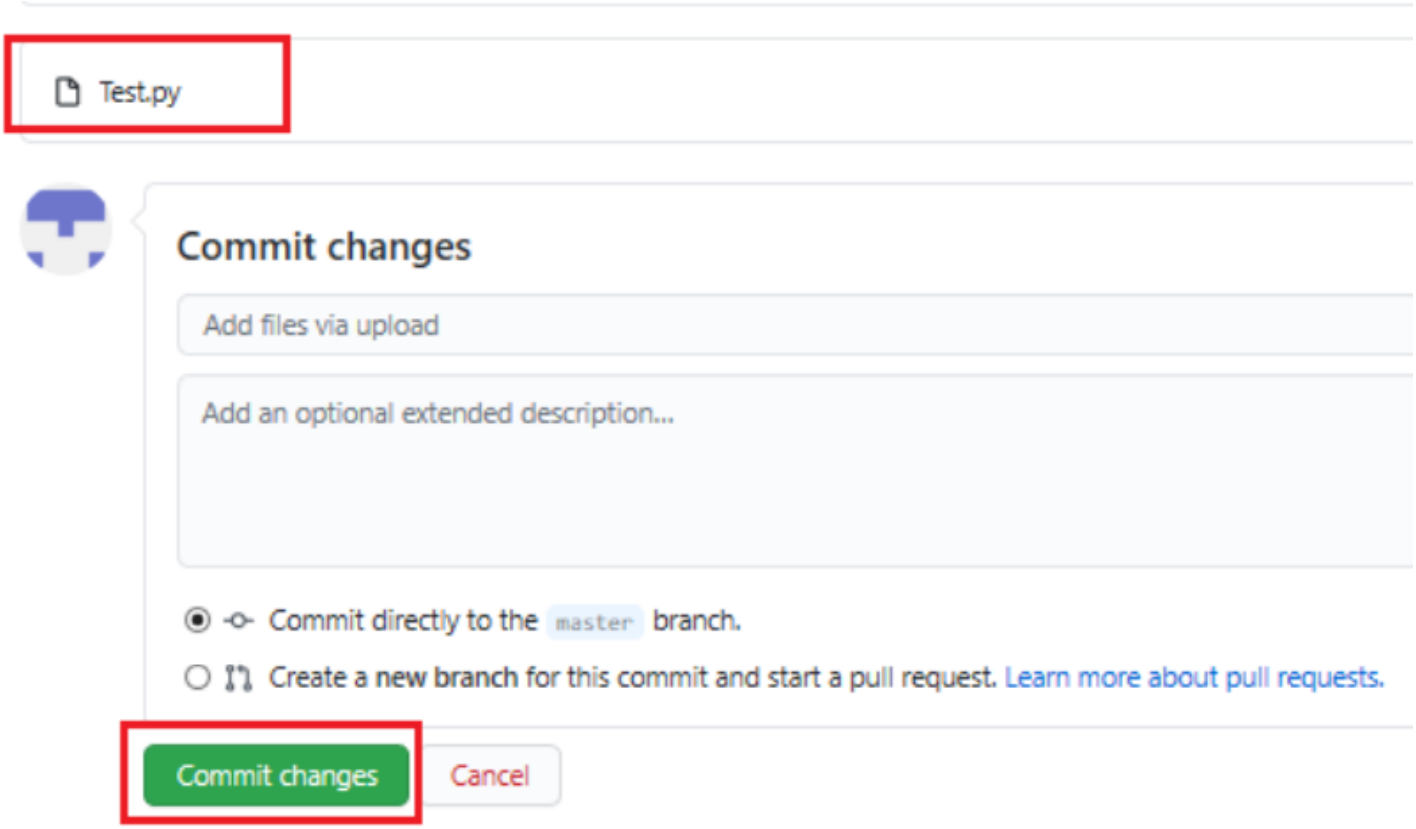
Step 1: Click `Add file` and select `Upload files` to upload a file (Upload any .txt,.ipynb, .png file) in the repository from the local computer.



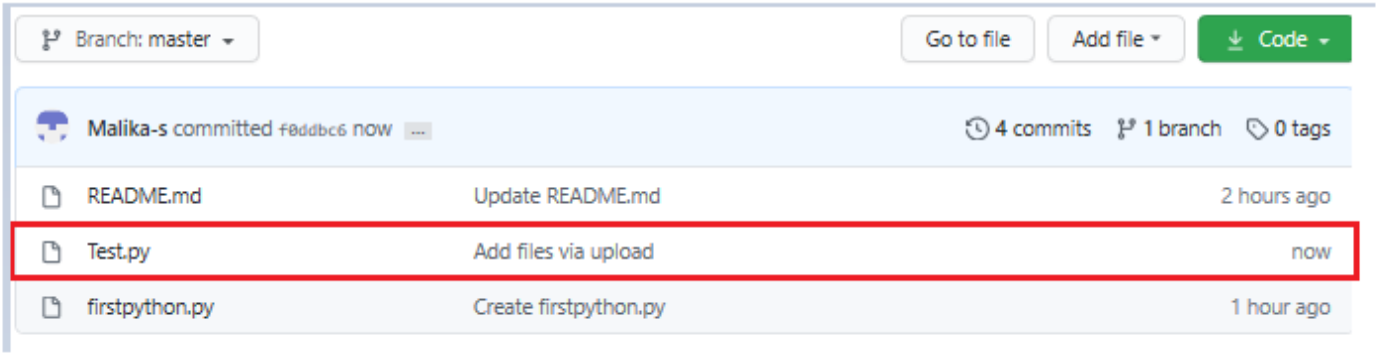
Step 2: Click on choose your files and choose any files from your computer.



Step 3: Once the file finishes uploading, click on `Commit` Changes



Step 4: Now, your file is uploaded in the repository.



1)How to Publish Notebook from Watson to Github

Go to the site below for video tutorial:

https://eu-de.dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/github-integration.html

token:

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

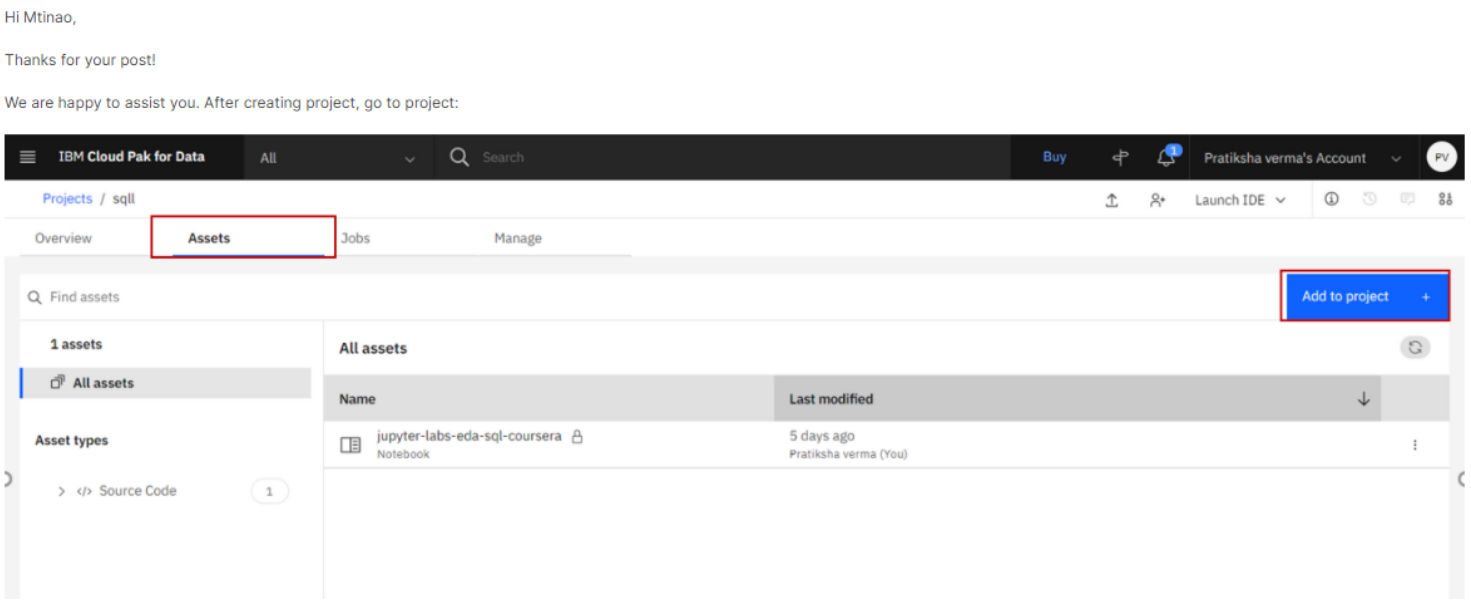
2) How to store code from terminal to Github

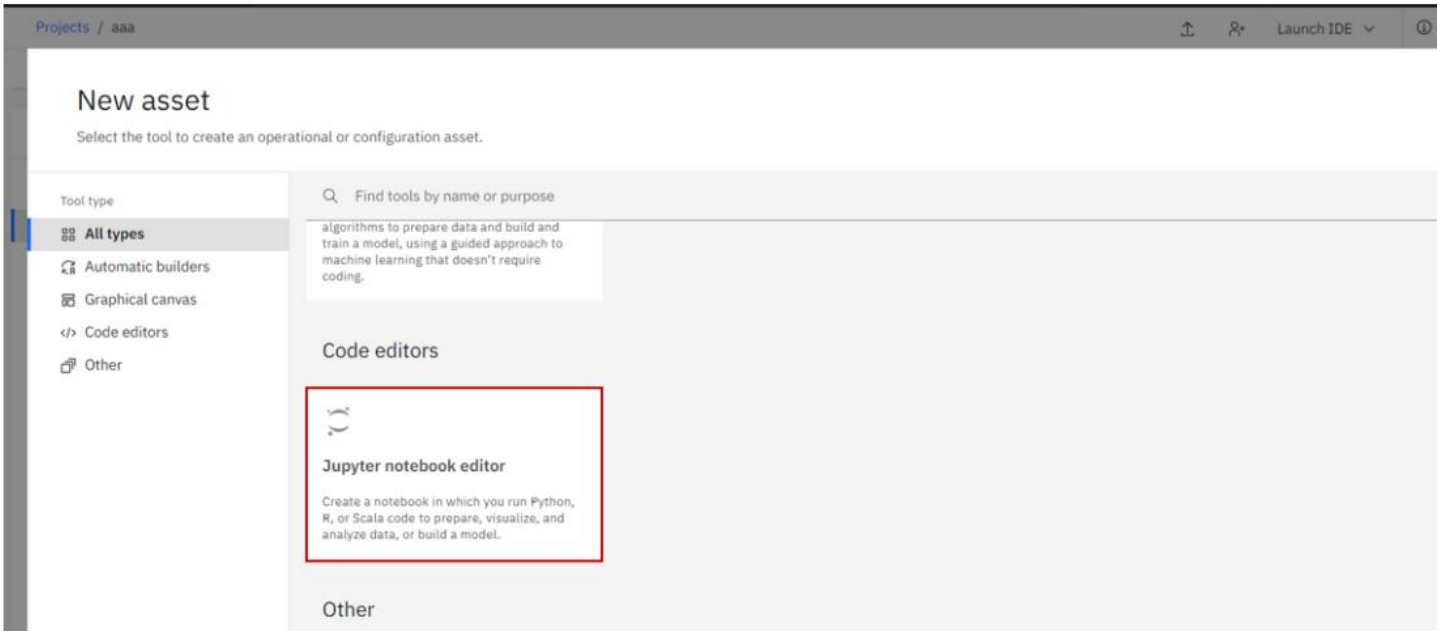
https://medium.com/@sauravbhagat_10426/how-to-upload-code-to-github-6db1c8ff56aa

1)How to Create and Share Jupyter Notebook to IBM Watson Studio

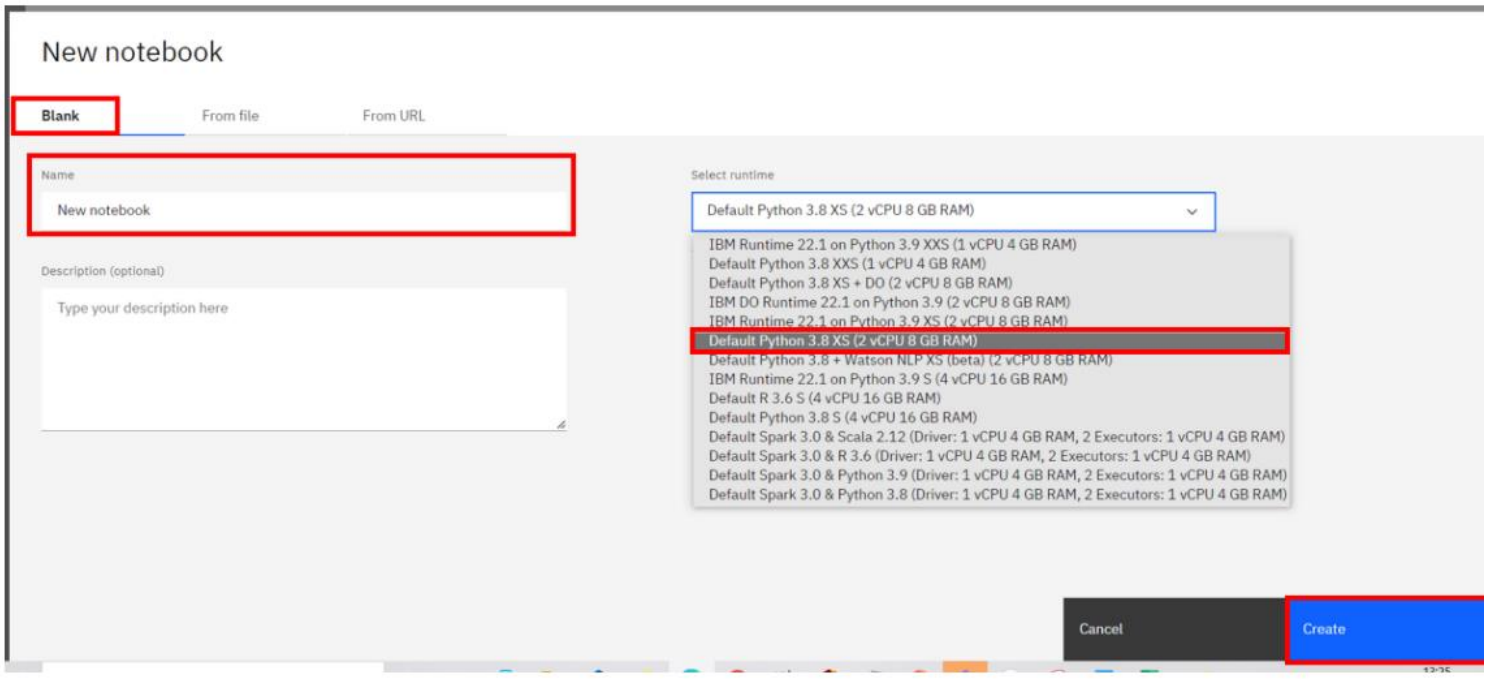
https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=notebooks-creating

https://eu-de.dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/creating-notebooks.html?audience=wdp





Click on asset ⟶new asset/add to project, then click on jupyter notebook.



After that if you want to add blank notebook then click on blank, give name to the new notebook and select the above python version as shown in screenshot.

If you want to add notebook url then select url tab, give name to notebook, add notebook url in notebook url box and select the above shown python version.

Collecting the Data

In this capstone assignment, we will be working with SpaceX launch data that is gathered from an API, specifically the SpaceX REST API.

This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome. Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not. The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
We have the different end points, for example: /capsules and /cores We will be working with the endpoint api.spacexdata.com/v4/launches/past.

Let's see how the API works.

We will use this URL to target a specific endpoint of the API to get past launch data. We will perform a get request using the requests library to obtain the launch data, which we will use to get the data from the API.
This result can be viewed by calling the .json() method. Our response will be in the form of a JSON, specifically a list of JSON objects. Since we are using an API, you will notice in the lab that when we get a response it is in the form of a JSON. Specifically, we have a list of JSON objects which each represent a launch.

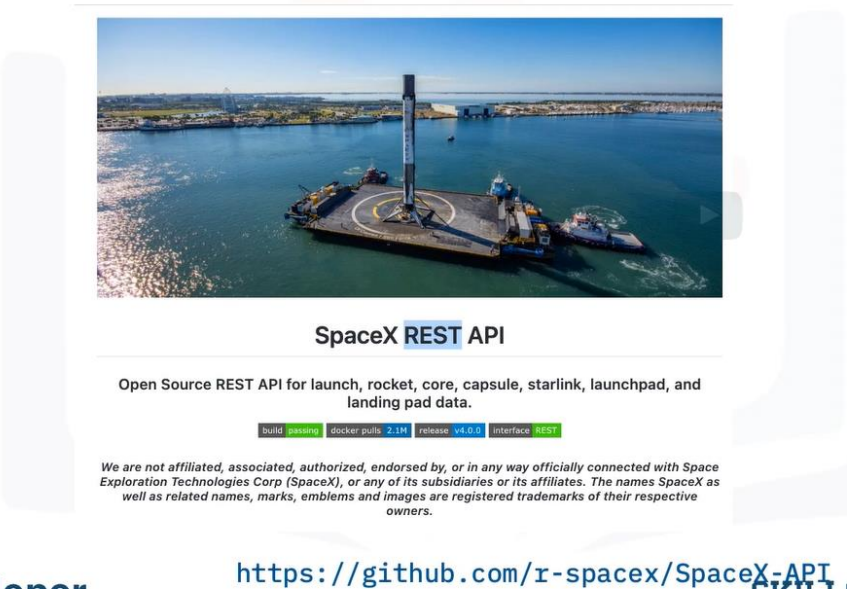
To convert this JSON to a dataframe, we can use the json_normalize function. This function will allow us to "normalize" the structured json data into a flat table. This is what your JSON will look like in a table form.
Another popular data source for obtaining Falcon 9 Launch data is web scraping related Wiki pages. In this lesson, you will be using the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then you need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis. We want to transform this raw data into a clean dataset which provides meaningful data on the situation we are trying to address: Wrangling Data using an API, Sampling Data, and Dealing with Nulls.

You will notice, in some of the columns, like rocket, we have an identification number, not actual data.
This means we will need to use the API again targeting another endpoint to gather specific data for each ID number. These functions are already created for you, and will use the following:
Booster, Launchpad, payload, and core.
The data will be stored in lists and will be used to create our dataset. Another issue we have is that the launch data we have includes data for the Falcon 1 booster whereas we only want falcon 9.
In this lab, you will need to figure out how to filter/sample the data to remove Falcon 1 launches.

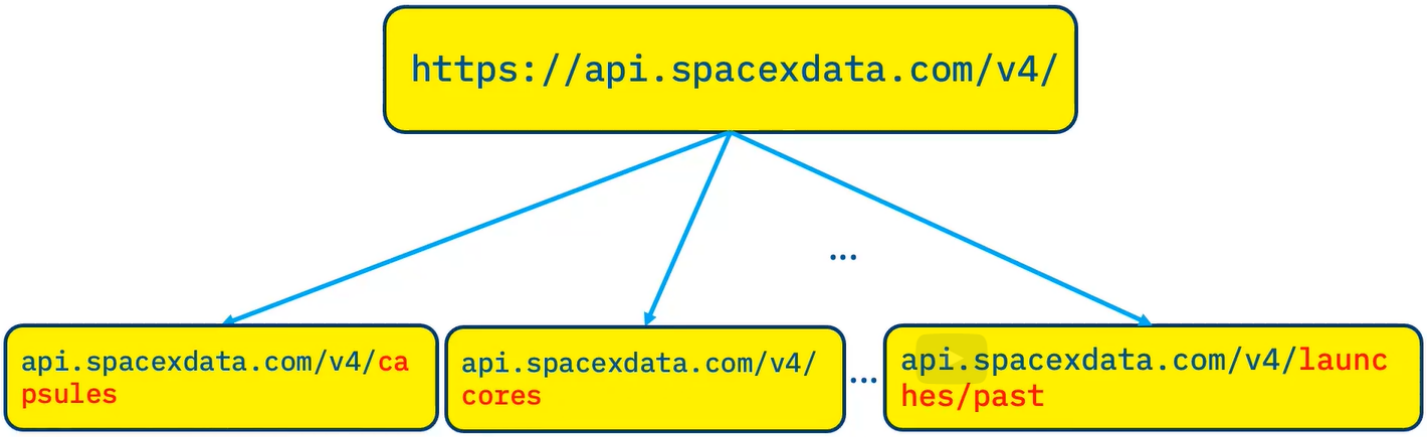
Finally, not all gathered data is perfect. We may end up with data that contains NULL values. We must sometimes deal with these null values in order to make the dataset viable for analysis. In this case, we will deal with the NULL values inside the PayloadMass. In this lab, you must figure out a way to calculate the mean of the PayloadMass data and then replace the null values in PayloadMass with the mean. We will leave the column LandingPad with NULL values, as it is represented when a landing pad is not used.
This will be dealt with using one hot encoding later on.

Collecting the Data

SpaceX REST API



This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome. Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.




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

response =requests.get(url)

response.json()
```

```
response.json()

[{'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/3c/0e/T8iJcSN3_o.png',
    'large': 'https://images2.imgbox.com/40/e3/GypSkayF_o.png'},
   'reddit': {'campaign': None,
    'launch': None,
    'media': None,
    'recovery': None},
   'flickr': {'small': [], 'original': []},
   'presskit': None,
   'webcast': 'https://www.youtube.com/watch?v=0a_00nJ_Y88',
   'youtube_id': '0a_00nJ_Y88',
   'article': 'https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html',
   'wikipedia': 'https://en.wikipedia.org/wiki/Demosat'},
  'static_fire_date_utc': '2006-03-17T00:00:00.000Z',
  'static_fire_date_unix': 1142553600,
  'tbd': False,
  'net': False,
  'window': 0,
  'rocket': '5e9d0d95eda69955f709d1eb',
  'success': False,
  'details': 'Engine failure at 33 seconds and loss of vehicle',
  'crew': [],
  'ships': [],
  'capsules': [],
  'payloads': ['5eb0e4b5b6c3bb0006eeb1e1'],
  'launchpad': '5e9e4502f5090995de566f86',
  'auto_update': True,
  'failures': [{'time': 33,
    'altitude': None,
    'reason': 'merlin engine failure'}],
  'flight_number': 1,
  'name': 'FalconSat',
  'date_utc': '2006-03-24T22:30:00.000Z',
  'date_unix': 1143239400,
  'date_local': '2006-03-25T10:30:00+12:00',
  'date_precision': 'hour',
  'upcoming': False,
  'cores': [{'core': '5e9e289df35918033d3b2623',
    'flight': 1,
    'gridfins': False,
    'legs': False,
    'reused': False,
    'landing_attempt': False,
    'landing_success': None,
    'landing_type': None,
    'landpad': None}],
  'id': '5eb87cd9ffd86e000604b32a'},
 {'fairings': {'reused': False,
   'recovery_attempt': False,
   'recovered': False,
   'ships': []},
  'links': {'patch': {'small': 'https://images2.imgbox.com/4f/e3/I0lkuJ2e_o.png',
    'large': 'https://images2.imgbox.com/be/e7/iNqsqVYM_o.png'},
   'reddit': {'campaign': None,
    'launch': None,
    'media': None,
    'recovery': None},
   'flickr': {'small': [], 'original': []},
   'presskit': None,
```

IBM Developer

We will perform a get request using the requests library to obtain the launch data, which we will use to get the data from the API. This result can be viewed by calling the .json() method. Our response will be in the form of a JSON, specifically a list of JSON objects.

Wrangling Data using an API

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

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	launchpad	auto_update	failures	flight_number	name	date_utc	date_unix
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86	True	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	1	FalconSat	2006-03-24T22:30:00.000Z	1143239400
1	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86	True	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	2	DemoSat	2007-03-21T01:10:00.000Z	1174439400
2	None	NaN	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Residual stage 1 thrust led to collision between stage 1 and stage 2	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]	5e9e4502f5090995de566f86	True	[{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to collision between stage 1 and stage 2'}]	3	Trailblazer	2008-08-03T03:34:00.000Z	1217734440
3	2008-09-20T00:00:00.000Z	1.221869e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	True	Ratsat was carried to orbit on the first successful orbital launch of any privately funded and developed, liquid-propelled carrier	[]	[]	[]	[5eb0e4b7b6c3bb0006eeb1e5]	5e9e4502f5090995de566f86	True	[]	4	RatSat	2008-09-28T23:15:00.000Z	1222643700

Specifically, we have a list of JSON objects which each represent a launch. To convert this JSON to a dataframe, we can use the json_normalize function. This function will allow us to “normalize” the structured json data into a flat table. This is what your JSON will look like in a table form.

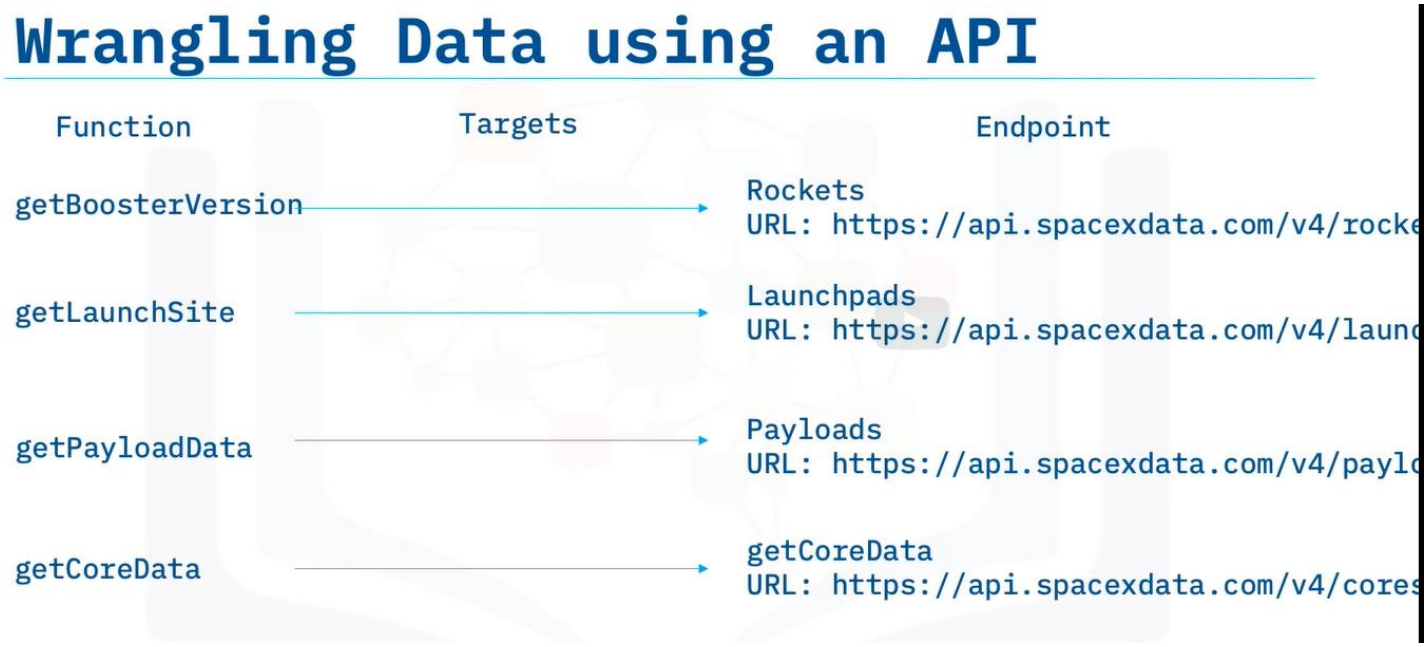
Web scraping Falcon 9 Launch records

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCAFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

you will be using the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then you need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis.

Data Wrangling Problems

- Wrangling Data using an API
- Sampling Data
- Dealing with Nulls



You will notice, in some of the columns, like rocket, we have an identification number, not actual data. This means we will need to use the API again targeting another endpoint to gather specific data for each ID number. These functions are already created for you, and will use the following: Booster, Launchpad, payload, and core. The data will be stored in lists and will be used to create our dataset.

Sampling Data

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCAFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

Another issue we have is that the launch data we have includes data for the Falcon 1 booster whereas we only want falcon 9. In this lab, you will need to figure out how to filter/sample the data to remove Falcon 1 launches.

Dealing with Nulls

Finally, not all gathered data is perfect. We may end up with data that contains NULL values. We must sometimes deal with these null values in order to make the dataset viable for analysis. In this case, we will deal with the NULL values inside the PayloadMass. In this lab, you must figure out a way to calculate the mean of the PayloadMass data and then replace the null values in PayloadMass with the mean. We will leave the column LandingPad with NULL values, as it is represented when a landing pad is not used. This will be dealt with using one hot encoding later on.

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/wiki/10.1.Introduction

10.1.Introduction

Seongjoo Brenden Song edited this page on Nov 7, 2021 · 2 revisions

In this capstone, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this module, you will be provided with an overview of the problem and the tools you need to complete the course.

Learning Objectives

- Use data science methodologies to define and formulate a real-world business problem.
- Use your data analysis tools to load a dataset, clean it, and find out interesting insights from it.

- Complete the Data Collection API Lab
- Data Collection with Web Scraping
- Data Wrangling

Watson Studio ym
Cloud Object Storage - bf
Python Basics for Data Science Project
https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/1.%20Complete%20the%20Data%20Collection%20API%20Lab.ipynb

Complete the Collection API Lab

SpaceX Falcon 9 first stage Landing Prediction

Lab 1: Collecting the data

Estimated time needed: **45** minutes

In this capstone, **we will predict if the Falcon 9 first stage will land successfully**. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.

Several examples of an unsuccessful landing are shown here:

Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.

Objectives

In this lab, you will make a get request to the SpaceX API. You will also do some basic data wrangling and formating.

- Request to the SpaceX API
- Clean the requested data

Import Libraries and Define Auxiliary Functions

We will import the following libraries into the lab

In [1]:
```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language,
# adding support for large, multi-dimensional arrays and matrices,
# along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime

# Setting this option will print all collumns of a dataframe
pd.set_option('display.max_columns', None)
# Setting this option will print all of the data in a feature
pd.set_option('display.max_colwidth', None)
```

Below we will define a series of helper 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.

In [2]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

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

In [3]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        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.

In [4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

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, and the serial of the core.

In [5]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
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'])
```

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

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [8]:
```
print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":
N3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":null,"launch"
all":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube
om/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/D
0:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb
e":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle"
["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat
x":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores"
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e450

	launchpad	flight_number	name	date_utc	date_unix	date_local	date_precision	upcoming	cores	auto_update	tbd
	e4502f5090995de566f86	1	FalconSat	2006-03-24T22:30:00.000Z	1143239400	2006-03-25T10:30:00+12:00	hour	False	[{'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	True	False

	tbd	launch_library_id	id	fairings.reused	fairings.recovery_attempt	fairings.recovered	fairings.ships	
	False	None	5eb87cd9ffd86e000604b32a	False	False	False	[]	https://images2.imgbo:

:	links.patch.small	links.patch.large	links.reddit.campaign	links.reddit.launch	links.reddit.media	links.reddit.recovery	links.flickr.small
	.imgbox.com/3c/0e/T8iJcSN3_o.png	https://images2.imgbox.com/40/e3/GypSkayF_o.png	None	None	None	None	[]

	links.patch.large	links.reddit.campaign	links.reddit.launch	links.reddit.media	links.reddit.recovery	links.flickr.small	links.flickr.original
	x.com/40/e3/GypSkayF_o.png	None	None	None	None	[]	[]

	links.presskit	links.webcast	links.youtube_id	
	None	https://www.youtube.com/watch?v=0a_00nJ_Y88	0a_00nJ_Y88	https://www.space.com/2falcon-1-r

1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e450
2	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 140, 'altitude': 35, 'reason': 'residual stage-1 thrust led to collision between stage 1 and stage 2'}]	Residual stage 1 thrust led to collision between stage 1 and stage 2	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e3, 5eb0e4b6b6c3bb0006eeb1e4]	5e9e450
	e4502f5090995de566f86	2	DemoSat	2007-03-21T01:10:00.000Z	1174439400	2007-03-21T13:10:00+12:00		hour	False	[{'core': '5e9e289ef35918416a3b2624', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	True	False			
3	2008-09-20T00:00:00.000Z	1.221869e+09	False	0.0	5e9d0d95eda69955f709d1eb	True	[]	Ratsat was carried to orbit on the first successful orbital launch of any privately funded and developed, liquid-propelled carrier rocket, the SpaceX Falcon 1	[]	[]	[]	[5eb0e4b7b6c3bb0006eeb1e5]	5e9e450

e4502f5090995de566f86	3	Trailblazer	2008-08-03T03:34:00.000Z	1217734440	2008-08-03T15:34:00+12:00	hour	False	[{'core': '5e9e289ef3591814873b2625', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	True	False				
4	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	True	[]	None	[]	[]	[]	[5eb0e4b7b6c3bb0006eeb1e6]	5e9e450	
e4502f5090995de566f86	4	RatSat	2008-09-28T23:15:00.000Z	1222643700	2008-09-28T11:15:00+12:00	hour	False	[{'core': '5e9e289ef3591855dc3b2626', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	True	False				
e4502f5090995de566f86	5	RazakSat	2009-07-13T03:35:00.000Z	1247456100	2009-07-13T15:35:00+12:00	hour	False	[{'core': '5e9e289ef359184f103b2627', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}]	True	False				

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now 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.

In [13]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have multiple payloads in a singl
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

- From the rocket we would like to learn the booster name
- From the payload we would like to learn the mass of the payload and the orbit that it is going to
- From the launchpad we would like to know the name of the launch site being used, the longitude, and the latitude.

- 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, whether the core is reused, whether 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, and the serial of the core.

The data from these requests will be stored in lists and will be used to create a new dataframe.

In [14]:
```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

These functions will apply the outputs globally to the above variables. Let's take a looks at BoosterVersion variable. Before we apply getBoosterVersion the list is empty:

In [15]:
```
BoosterVersion
```

Out[15]:
```
[]
```

Now, let's apply `getBoosterVersion` function method to get the booster version

In [16]:
```
# Call getBoosterVersion
getBoosterVersion(data)
```

the list has now been update

In [17]:
```
BoosterVersion[0:5]
```

Out[17]:
```
['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

In [18]:
```
# Call getLaunchSite
getLaunchSite(data)
```

In [19]:
```
# Call getPayloadData
getPayloadData(data)
```

In [20]:
```
# Call getCoreData
getCoreData(data)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

In [21]:
```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

In [25]:
```
# Create a data from launch_dict
launch_df = pd.DataFrame.from_dict(launch_dict)
```

Show the summary of the dataframe

In [26]:
```
# Show the head of the dataframe
launch_df.head()
```

Out[26]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366

Latitude
9.047721
9.047721
9.047721
9.047721
28.561857

Task 2: Filter the dataframe to only include Falcon 9 launches

Finally we will 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.

In [32]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] == 'Falcon 9']
```

Now that we have removed some values we should reset the FlgihtNumber column

In [33]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/pandas/core/indexing.py:966: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

self.obj[item] = s

Out[33]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True

90 rows × 17 columns

LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
None	1.0	0	B0003	-80.577366	28.561857
None	1.0	0	B0005	-80.577366	28.561857
None	1.0	0	B0007	-80.577366	28.561857
None	1.0	0	B1003	-120.610829	34.632093
None	1.0	0	B1004	-80.577366	28.561857
...	...	...	...	...	...
5e9e3032383ecb6bb234e7ca	5.0	7	B1060	-80.603956	28.608058
5e9e3032383ecb6bb234e7ca	5.0	7	B1058	-80.603956	28.608058
5e9e3032383ecb6bb234e7ca	5.0	9	B1051	-80.603956	28.608058
5e9e3033383ecbb9e534e7cc	5.0	7	B1060	-80.577366	28.561857
5e9e3032383ecb6bb234e7ca	5.0	2	B1062	-80.577366	28.561857

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

```
In [34]: data_falcon9.isnull().sum()
```

```
Out[34]: FlightNumber      0
         Date              0
         BoosterVersion    0
         PayloadMass       5
         Orbit             0
         LaunchSite        0
         Outcome           0
         Flights           0
         GridFins          0
         Reused            0
         Legs              0
         LandingPad       26
         Block             0
         ReusedCount       0
         Serial            0
         Longitude         0
         Latitude          0
         dtype: int64
```

Before we can continue we must deal with these missing values. The LandingPad column will retain None values to

represent when landing pads were not used.

Task 3: Dealing with Missing Values

Calculate below the mean for the PayloadMass using the .mean(). Then use the mean and the .replace() function to replace np.nan values in the data with the mean you calculated

In [41]:
```
# Calculate the mean value of PayloadMass column
PayloadMass_mean = data_falcon9.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PayloadMass_mean)
```

/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/ipykernel/__main__.py:4: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

You should see the number of missing values of the PayLoadMass change to zero.

In [42]:
```
data_falcon9.isnull().sum()
```

Out[42]:
```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

Now we should have no missing values in our dataset except for in LandingPad.

We can now export it to a **CSV** for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

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

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/2.%20Data%20Collection%20with%20Web%20Scraping.ipynb

Data Collection with Web Scraping

Space X Falcon 9 First Stage Landing Prediction

Web scraping Falcon 9 and Falcon Heavy Launches Records from Wikipedia

Estimated time needed: **40** minutes

In this lab, you will be performing web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches

https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Falcon 9 first stage will land successfully

Several examples of an unsuccessful landing are shown here:

More specifically, the launch records are stored in a HTML table shown below:

Objectives

Web scrap Falcon 9 launch records with BeautifulSoup:

- Extract a Falcon 9 launch records HTML table from Wikipedia
- Parse the table and convert it into a Pandas data frame

First let's import required packages for this lab

In [1]:
```
!pip3 install beautifulsoup4
!pip3 install requests
```

```
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7
d, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7
```

In [2]:
```
import sys

import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

and we will provide some helper functions for you to process web scraped HTML table

In [3]:

```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out
```

```
def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the List of Falcon 9 and Falcon Heavy launches Wikipage updated on 9th June 2021

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

In [8]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

In [9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data, 'html5lib')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

In [13]:
```
# Use soup.title attribute
soup.title
```

Out[13]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, 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. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards the end of this lab

In [14]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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

In [17]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version
="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
```

You should able to see the columns names embedded in the table header elements <th> as follows:

```
<tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,
<br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12">[c]</a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
</th>
<th scope="col">Customer
</th>
<th scope="col">Launch<br/>outcome
</th>
<th scope="col"><a href="/wiki/Falcon_9_first-stage_landing_tests" title="Falcon 9 first-stage landing
tests">Booster<br/>landing</a>
</th></tr>
```

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

In [21]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if(name != None and len(name) > 0):
        column_names.append(name)
```

Check the extracted column names

In [37]:
```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

In [38]:
```
launch_dict= dict.fromkeys(column_names)

# 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']=[]
```

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, we have provided an incomplete code snippet below to help you to fill up the launch_dict. Please complete the following code snippet with TODOs or you can choose to write your own logic to parse all launch tables:

In [39]:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            print(date)
            launch_dict['Date'].append(date)


            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            print(time)
            launch_dict['Time'].append(time)
```

```
    # Booster version
    # TODO: Append the bv into launch_dict with key `Version Booster`
    bv=booster_version(row[1])
    if not(bv):
        bv=row[1].a.string
    print(bv)
    launch_dict['Version Booster'].append(bv)

    # Launch Site
    # TODO: Append the bv into launch_dict with key `Launch Site`
    launch_site = row[2].a.string
    print(launch_site)
    launch_dict['Launch site'].append(launch_site)

    # Payload
    # TODO: Append the payload into launch_dict with key `Payload`
    payload = row[3].a.string
    print(payload)
    launch_dict['Payload'].append(payload)

    # Payload Mass
    # TODO: Append the payload_mass into launch_dict with key `Payload mass`
    payload_mass = get_mass(row[4])
    print(payload)
    launch_dict['Payload mass'].append(payload_mass)

    # Orbit
    # TODO: Append the orbit into launch_dict with key `Orbit`
    orbit = row[5].a.string
    print(orbit)
    launch_dict['Orbit'].append(orbit)

    # Customer
    # TODO: Append the customer into launch_dict with key `Customer`
    customer = row[6].text.strip()
    print(customer)
    launch_dict['Customer'].append(customer)
```

```
# Launch outcome
# TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
launch_outcome = list(row[7].strings)[0]
print(launch_outcome)
launch_dict['Launch outcome'].append(launch_outcome)

# Booster landing
# TODO: Append the launch_outcome into launch_dict with key `Booster landing`
booster_landing = landing_status(row[8])
print(booster_landing)
launch_dict['Booster landing'].append(booster_landing)
```

```
1
4 June 2010
18:45
F9 v1.0B0003.1
CCAFS
Dragon Spacecraft Qualificat
Dragon Spacecraft Qualificat
LEO
SpaceX
Success

Failure
2
8 December 2010
15:43
F9 v1.0B0004.1
CCAFS
Dragon
Dragon
LEO
NASA (COTS)
NRO
Success
Failure
3
22 May 2012
07:44
F9 v1.0B0005.1
CCAFS
Dragon
Dragon
LEO
NASA (COTS)
Success
No attempt

4
8 October 2012
00:35
F9 v1.0B0006.1
CCAFS
SpaceX CRS-1
SpaceX CRS-1
LEO
NASA (CRS)
```

```
Success
121
6 June 2021
04:26
F9 B5
CCSFS
SXM-8
SXM-8
GTO
Sirius XM
Success

Success
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
In [40]: df=pd.DataFrame(launch_dict)
```

```
In [41]: df
```

Out[41]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX Capella Space and Tyvak	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA (CRS)	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

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

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/3.%20Complete%20the%20EDA%20lab.ipynb

Check Points: Rest API

Bookmarked

Question 1

1/1 point (ungraded)

Did you Request and parse the SpaceX launch data using the GET request?

- (●) Yes
- ( ) No

✔

Submit

Question 2

1/1 point (ungraded)

Did you filter the dataframe to only include Falcon 9 launches?

- (●) Yes
- ( ) No

✔

Submit

Question 3

1/1 point (ungraded)

Did you relace None values in the PayloadMass with the mean?

- (●) Yes
- ( ) No

✔

Submit

Data Wrangling Overview

In this video, we will review data wrangling.

Let's review some of the attributes: Flight Number, Date, Booster version, Payload mass Orbit, Launch Site,

Outcome: this is the status of the first stage Flights, Grid Fins: these help with landing Reused, Legs: used in landing Landing pad, Block, Reused count, Serial, Longitude and latitude of launch

Let's take a look at some of these attributes.

The column "LaunchSite" contains the different launch sites, including:

Vandenberg AFB Space Launch

Kennedy Space Center

CCAFS SLC 40

The column orbits are the different orbits of the payload.

For Example:

LEO: Low Earth orbit (LEO)is an Earth-centered orbit with an altitude of 2,000 km

GTO A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. It is located at 22,236 miles (35,786 kilometers) above Earth's equator.

The column Outcome indicates if the first stage successfully landed.

There are 8 of them, for example.

True ASDS means the booster successfully landed to a drone ship as shown the following looped video.

False ASDS means the mission outcome was unsuccessfully landed to a drone ship as shown in the video loop.

Outcome

We would like landing outcomes to be converted to Classes

y. y.

(either 0 or 1).

0 is a bad outcome, that is, the booster did not land.

1 is a good outcome, that is, the booster did land.

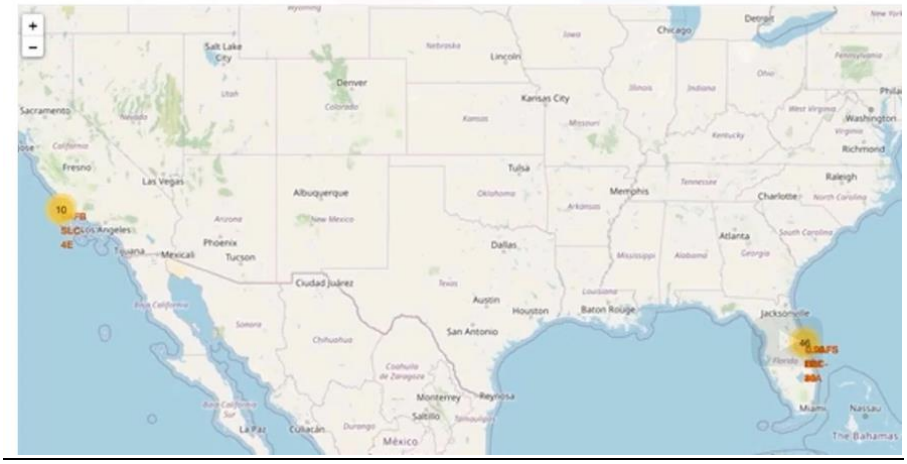
The variable Y will represent the classification variable that represents the outcome of each launch.

Data Wrangling

Exploratory Data Analysis

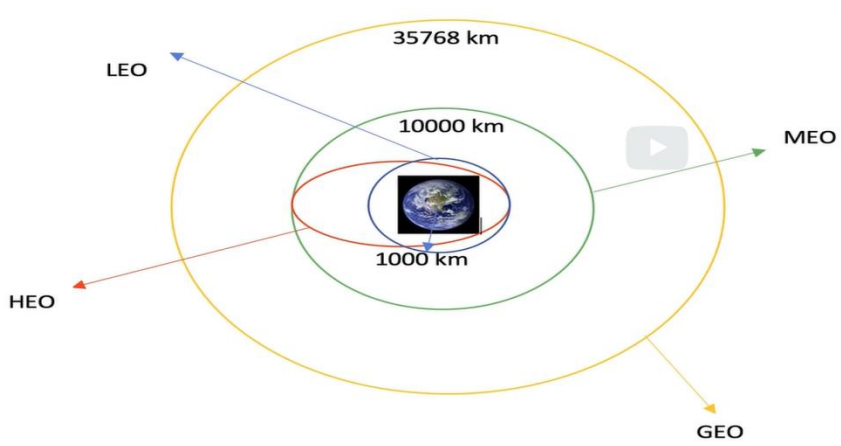
Overview of Dataset

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857



The column “LaunchSite” contains the different launch sites, including: Vandenberg AFB Space Launch Kennedy Space Center CCAFS SLC 40

Orbit



LEO: Low Earth orbit (LEO)is an Earth-centered orbit with an altitude of 2,000 km GTO A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. It is located at 22,236 miles (35,786 kilometers) above Earth's equator.

Outcome

- The column Outcome indicates if the first stage successfully landed

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

True ASDS means the booster successfully landed to a drone ship as shown the following looped video.

False ASDS means the mission outcome was unsuccessfully landed to a drone ship as shown in the video loop.

Outcome

- We would like landing outcomes to be converted to Classes *y*.(either 0 or 1).
- 0 is a bad outcome i.e., the booster did not land
- 1 is a good outcome i.e., the booster did land

The variable Y will represent the classification variable that represents the outcome of each launch.

Data Wrangling

Space X Falcon 9 First Stage Landing Prediction

Lab 2: Data wrangling

Estimated time needed: **60** minutes

In this lab, we will perform some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

In this lab we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Falcon 9 first stage will land successfully

Several examples of an unsuccessful landing are shown here:

Objectives

Perform exploratory Data Analysis and determine Training Labels

- Exploratory Data Analysis
- Determine Training Labels

Import Libraries and Define Auxiliary Functions

We will import the following libraries.

In [1]:
```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collecti
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

In [2]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

Out[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0

ReusedCount	Serial	Longitude	Latitude
0	B0003	-80.577366	28.561857
0	B0005	-80.577366	28.561857
0	B0007	-80.577366	28.561857
0	B1003	-120.610829	34.632093
0	B1004	-80.577366	28.561857
0	B1005	-80.577366	28.561857
0	B1006	-80.577366	28.561857
0	B1007	-80.577366	28.561857
0	B1008	-80.577366	28.561857
0	B1011	-80.577366	28.561857

Identify and calculate the percentage of the missing values in each attribute

In [3]:
```
df.isnull().sum()/df.count()*100
```

Out[3]:
```
FlightNumber      0.000
Date              0.000
BoosterVersion    0.000
PayloadMass       0.000
Orbit             0.000
LaunchSite        0.000
Outcome           0.000
Flights           0.000
GridFins          0.000
Reused            0.000
Legs              0.000
LandingPad       40.625
Block             0.000
ReusedCount       0.000
Serial            0.000
Longitude         0.000
Latitude          0.000
dtype: float64
```

Identify which columns are numerical and categorical:

In [4]:
```
df.dtypes
```

Out[4]:
```
FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
dtype: object
```

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column LaunchSite

Next, let's see the number of launches for each site.

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

In [6]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[6]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO \[1].

- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation\[2].

- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website \[3] .

- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .

- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth \[5] .

- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth \[6].

- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) \[7]

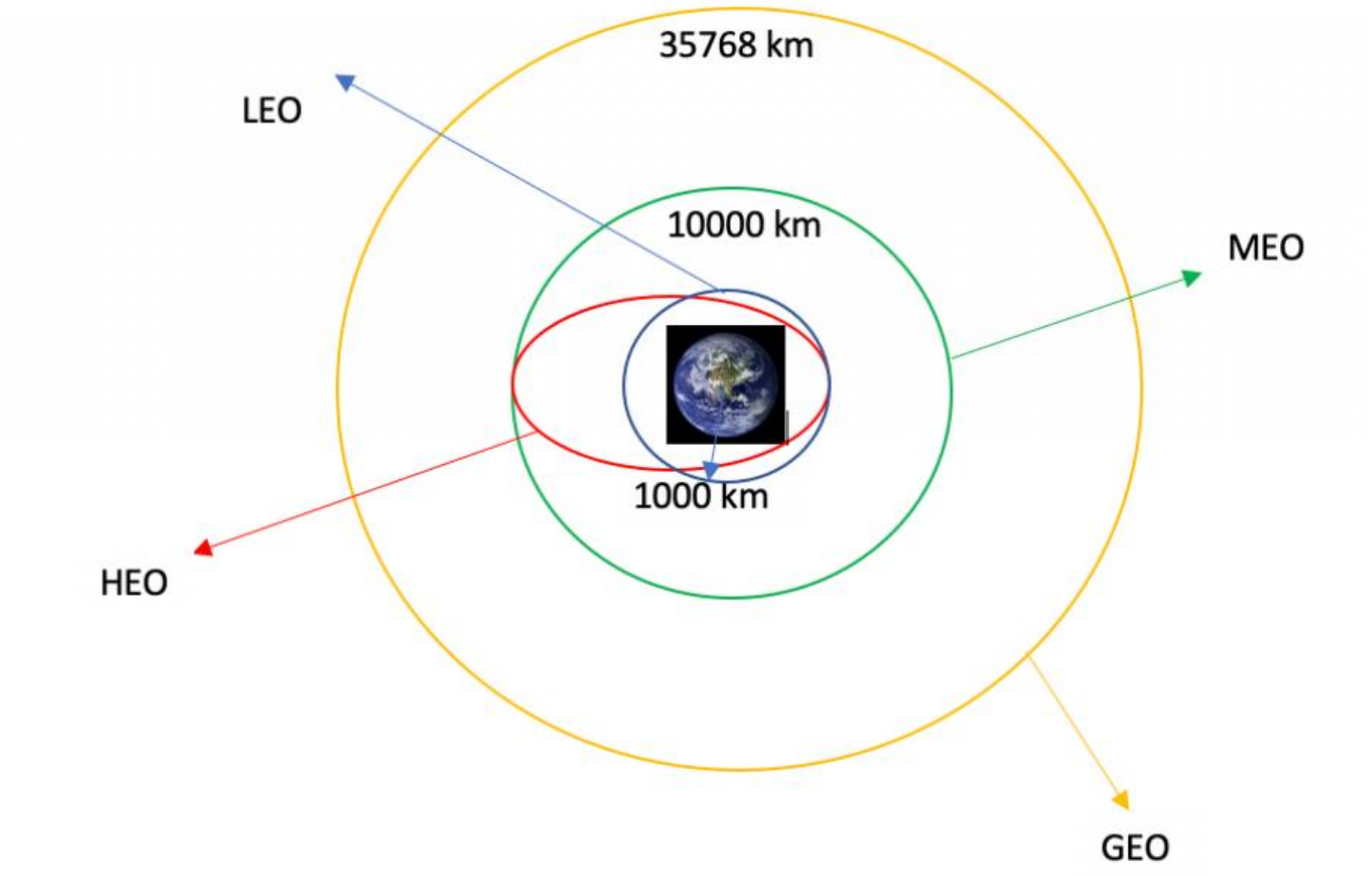
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours \[8]

- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) \[9]

- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's rotation \[10]

- **PO** It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth \[11]

some are shown in the following plot:



TASK 2: 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

```
In [7]:  # Apply value_counts on Orbit column
         df.Orbit.value_counts()
```

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

TASK 3: 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.

In [22]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

Out[22]:
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

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

In [9]:
```
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 None ASDS
6 False Ocean
7 False RTLS
```

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

In [10]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[10]:
```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

TASK 4: 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:

In [20]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df.Outcome:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will 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

In [21]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[21]:

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

In [17]:
```
df.head(5)
```

Out[17]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False

LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
NaN	1.0	0	B0003	-80.577366	28.561857	0
NaN	1.0	0	B0005	-80.577366	28.561857	0
NaN	1.0	0	B0007	-80.577366	28.561857	0
NaN	1.0	0	B1003	-120.610829	34.632093	0
NaN	1.0	0	B1004	-80.577366	28.561857	0

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

In [18]:
```
df["Class"].mean()
```

Out[18]:
```
0.6666666666666666
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

df.to_csv("dataset_part_2.csv", index=False)

Authors

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Nayef Abou Tayoun is a Data Scientist at IBM and pursuing a Master of Management in Artificial intelligence degree at Queen's University.

Check Points: Data Wrangling

Bookmarked

Question 1

1/1 point (ungraded)

Did you calculate the number of launches on each site?

- (●) Yes
- ( ) No

✔

Submit

Question 2

1/1 point (ungraded)

Did you calculate the number and occurrence of each orbit?

- (●) Yes
- ( ) No

✔

Submit

Question 3

1/1 point (ungraded)

Did you calculate the number and occurrence of mission outcome per orbit type?

- (●) Yes
- ( ) No

✔

Submit

Question 4

1/1 point (ungraded)

Did you create a landing outcome label from outcome column?

- (●) Yes
- ( ) No

✔

Submit

Graded Quiz: Data Wrangling

Bookmarked

Quiz due Mar 26, 2022 15:06 +08 Completed

Question 1

1/1 point (graded)

How many launches came from **CCAFS SLC 40**?

55 ✔

55

Submit You have used 1 of 2 attempts

Question 2

1/1 point (graded)

What was the success rate?

- ○ 80%
- ○ 40%
- ◉ 67% ✔

Answer
Correct: You can specify optional feedback like this, which appears after this answer is submitted.

Submit You have used 1 of 2 attempts Save Show answ

Question 3

1/1 point (graded)

In the lab you used the method .value_counts() to determine the number and occurrence of each orbit in the column Orbit. What was the value for Orbit with the column name GTO.

27 ✔

27

Submit You have used 1 of 2 attempts Save Show answ

Question 4

1/1 point (graded)

How many landing outcomes in the column landing_outcomes had a value of none

19

19

Submit You have used 1 of 2 attempts

10.2.Exploratory Data Analysis (EDA)

Seongjoo Brenden Song edited this page on Nov 7, 2021 · 2 revisions

In this module, you will collect data on the Falcon 9 first-stage landings. You will use a RESTful API and web scraping. You will also convert the data into a dataframe and then perform some data wrangling.

Learning Objectives

- Use your data visualization skills to visualize the data and extract meaningful patterns to guide the modeling process.

- Complete the EDA with SQL
- Complete the EDA with Visualization lab

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/4.%20Complete%20the%20EDA%20with%20SQL%20lab.ipynb

Module Introduction and Learning Objectives

Module Introduction

In this module, you will collect data on the Falcon 9 first-stage landings. You will use a RESTful API and web scraping. You will also convert the data into a dataframe and then perform some data wrangling.

Learning Objectives

- Use your data visualization skills to visualize the data and extract meaningful patterns to guide the modeling process.
- Use your data analysis tools to load a dataset, clean it, and find out interesting insights from it.

Exploratory Data Analysis

Exploratory Data Analysis is the first step of any data science project.

In the first lab, you will perform some Exploratory Data Analysis using a database. In the second lab, you will see if the data can be used to automatically determine if the Falcon 9's second stage will land.

Some attributes can be used to determine if the first stage can be reused. We can then use these features with machine learning to automatically predict if the first stage can land successfully, for example. You can observe that the success rate since 2013 has improved. We can incorporate this as a feature via launch Number.

We see that different launch sites have different success rates. As a result, they can be used to help determine if the first stage will land successfully. CCAFS LC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E have a success rate of around 77%.

Combining attributes also gives us more information.

If we overlay the result of the landing outcomes as a color we see that CCAFS LC-40, has a success rate of 60%, but if the mass is above 10,000 kg the success rate is 100%.

Therefore, we will combine multiple features. In the lab, you will determine what attributes are correlated with successful landings. The categorical variables will be converted using one hot encoding, preparing the data for a machine learning model that will predict if the first stage will successfully land.

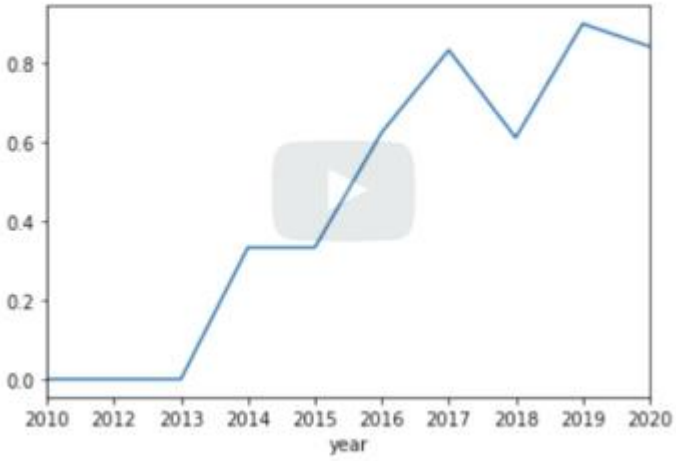
Exploratory Data Analysis

Exploring the Data

- In the first lab you will perform some Exploratory Data Analysis using a database

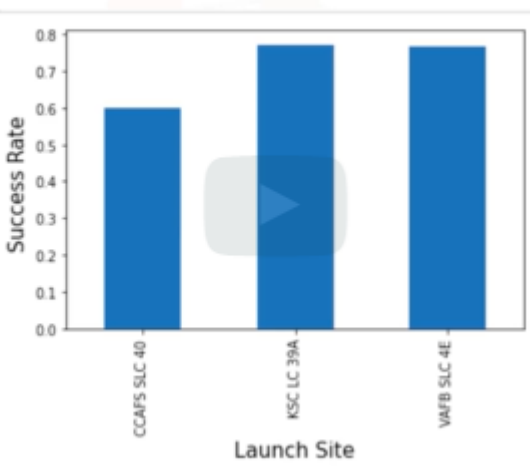
In the first lab, you will perform some Exploratory Data Analysis using a database. In the second lab, you will see if the data can be used to automatically determine if the Falcon 9's second stage will land.

Feature Engineering

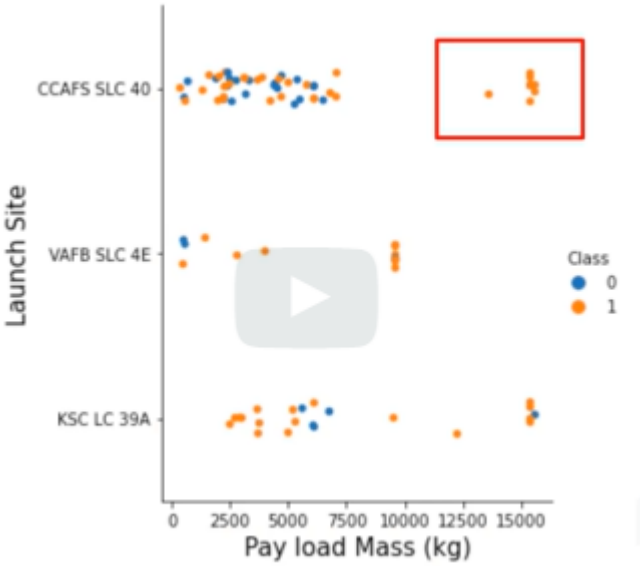


Some attributes can be used to determine if the first stage can be reused. We can then use these features with machine learning to automatically predict if the first stage can land successfully, for example. You can observe that the success rate since 2013 has improved. We can incorporate this as a feature via launch Number.

Feature Engineering



We see that different launch sites have different success rates. As a result, they can be used to help determine if the first stage will land successfully. CCAFS LC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E have a success rate of around 77%.



Combining attributes also gives us more information. If we overlay the result of the landing outcomes as a color

we see that CCAFS LC-40, has a success rate of 60%, but if the mass is above 10,000 kg the success rate is 100%.

Therefore, we will combine multiple features.

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062	GridFins_False	GridFins_True	Reused_False	Reused_True	Legs_False	Legs_True
0	1	6104.959412	1	1.0	0	0	0	0	0	0	...	0	0	0	0	1	0	1	0	1	0
1	2	525.000000	1	1.0	0	0	0	0	0	0	...	0	0	0	0	1	0	1	0	1	0
2	3	677.000000	1	1.0	0	0	0	0	0	1	...	0	0	0	0	1	0	1	0	1	0
3	4	500.000000	1	1.0	0	0	0	0	0	0	...	0	0	0	0	1	0	1	0	1	0
4	5	3170.000000	1	1.0	0	0	0	1	0	0	...	0	0	0	0	1	0	1	0	1	0

5 rows × 83 columns

In the lab, you will determine what attributes are correlated with successful landings. The categorical variables will be converted using one hot encoding, preparing the data for a machine learning model that will predict if the first stage will successfully land

Locating your Service Credentials

https://cloud.ibm.com/docs/Cloudant?topic=Cloudant-locating-your-service-credentials

Complete EDA with SQL

Assignment: SQL Notebook for Peer Assignment

Estimated time needed: **60** minutes.

Introduction

Using this Python notebook you will:

1. Understand the Spacex DataSet
2. Load the dataset into the corresponding table in a Db2 database
3. Execute SQL queries to answer assignment questions

Overview of the DataSet

SpaceX has gained worldwide attention for a series of historic milestones.

It is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars wheras other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage.

Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

This dataset includes a record for each payload carried during a SpaceX mission into outer space.

Download the datasets

This assignment requires you to load the spacex dataset.

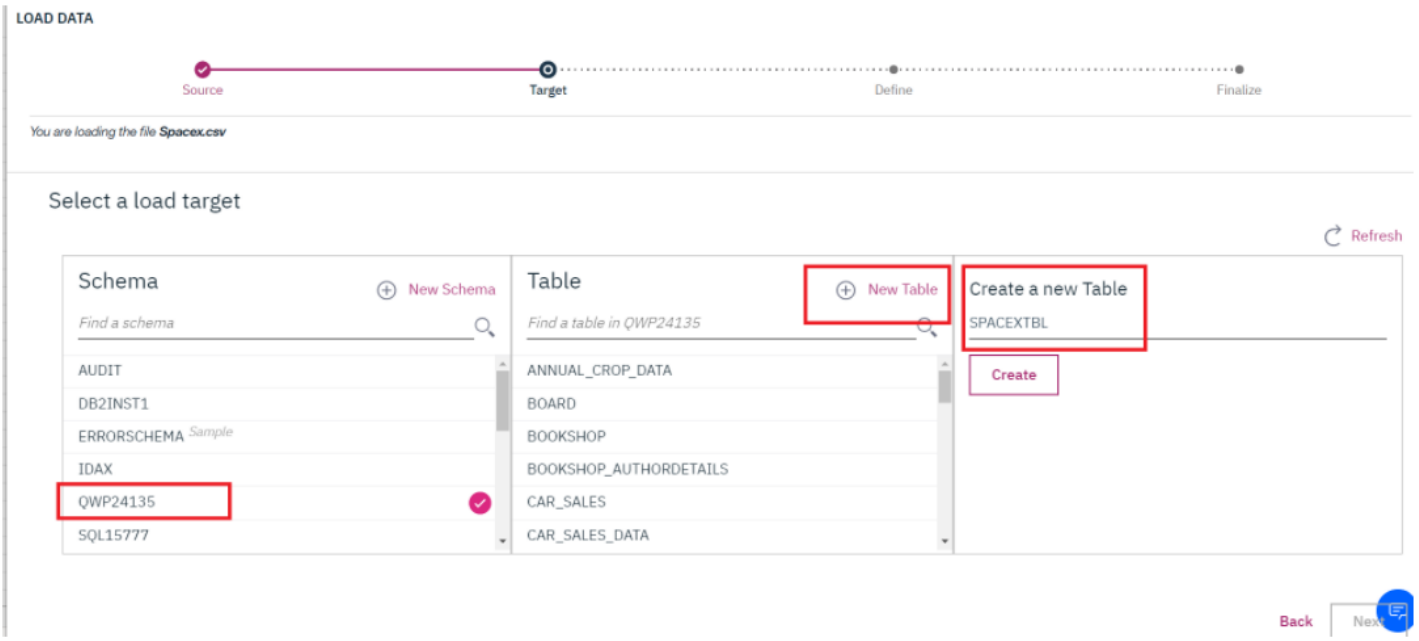
In many cases the dataset to be analyzed is available as a .CSV (comma separated values) file, perhaps on the internet. Click on the link below to download and save the dataset (.CSV file):

Spacex DataSet

Store the dataset in database table

it is highly recommended to manually load the table using the database console LOAD tool in DB2.

it is highly recommended to manually load the table using the database console LOAD tool in DB2.



Now open the Db2 console, open the LOAD tool, Select / Drag the .CSV file for the dataset, Next create a New Table, and then follow the steps on-screen instructions to load the data. Name the new table as follows:

SPACEXDATASET

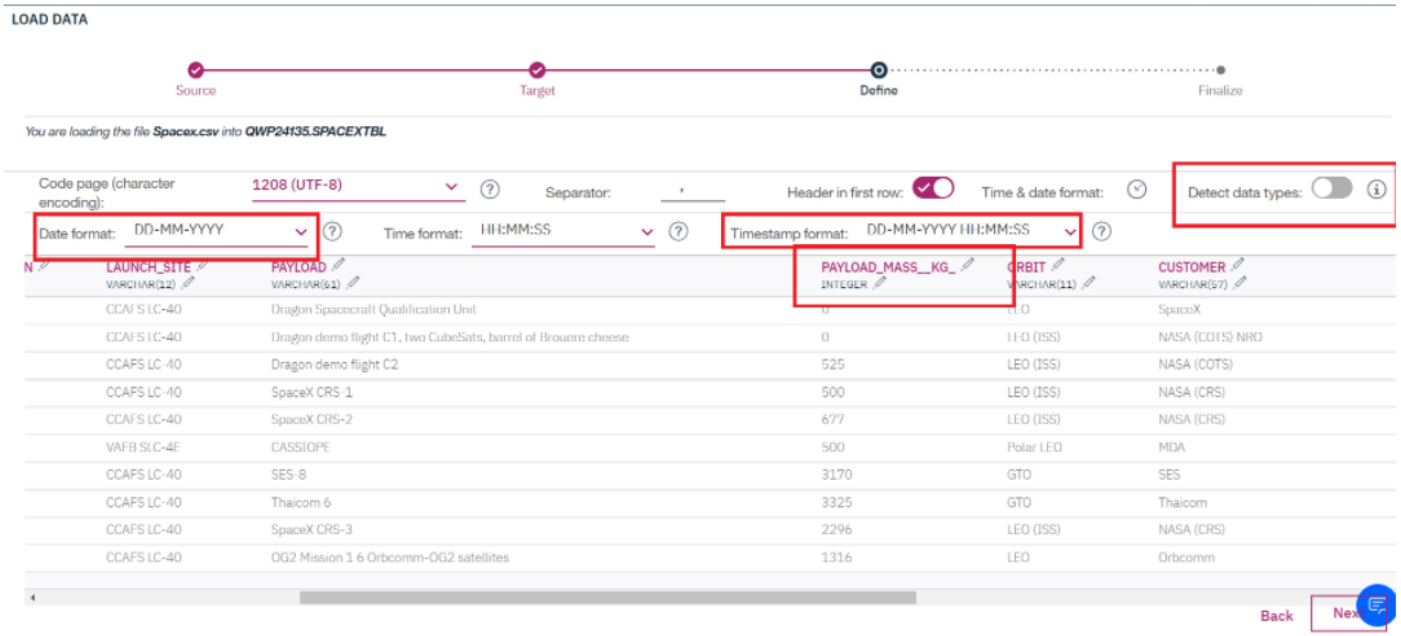
Follow these steps while using old DB2 UI which is having Open Console Screen

Note:While loading Spacex dataset, ensure that detect datatypes is disabled. Later click on the pencil icon(edit option).

1. Change the Date Format by manually typing DD-MM-YYYY and timestamp format as DD-MM-YYYY HH\:MM:SS.

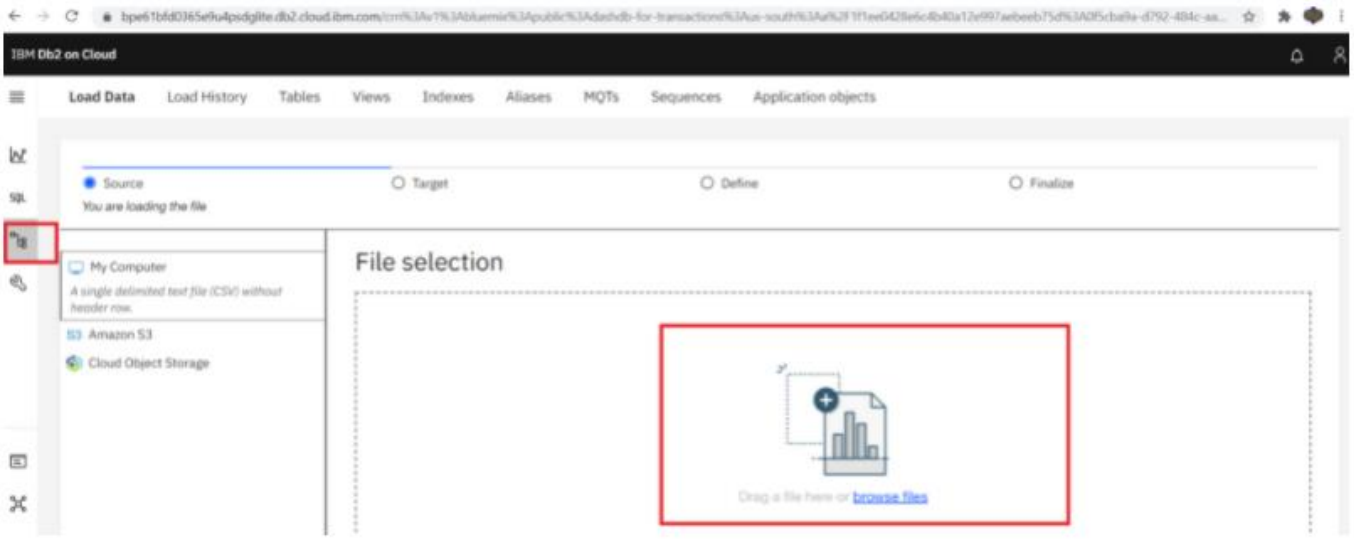
Here you should place the cursor at Date field and manually type as DD-MM-YYYY.

2. Change the PAYLOAD*MASS*_KG_ datatype to INTEGER.

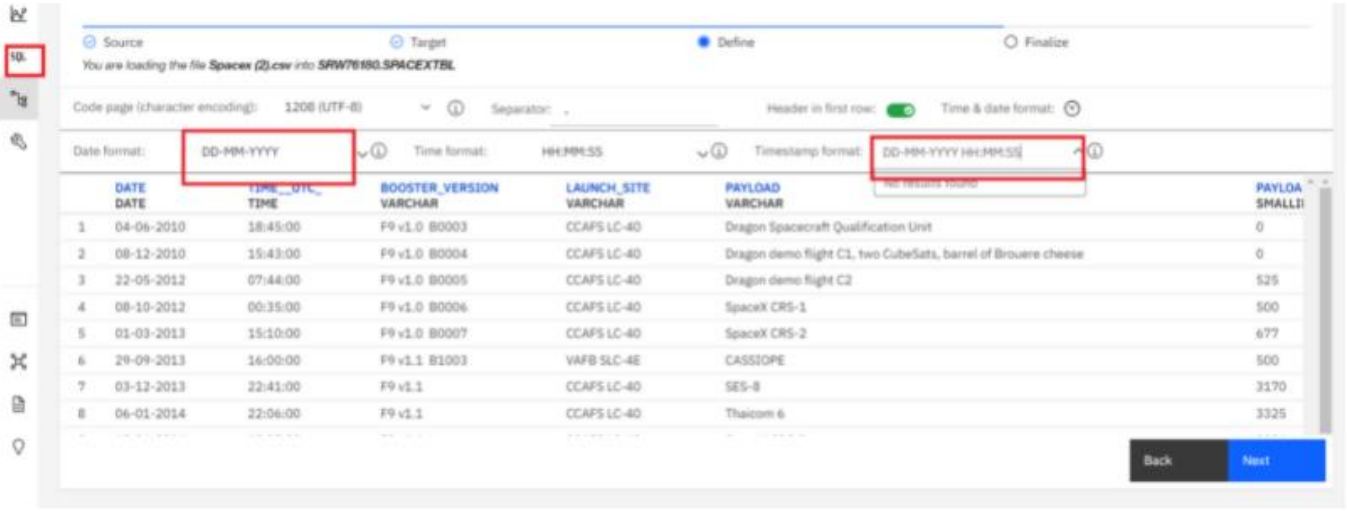


Changes to be considered when having DB2 instance with the new UI having Go to UI screen

- Refer to this insruction in this link for viewing the new Go to UI screen.
- Later click on **Data link(below SQL)** in the Go to UI screen and click on **Load Data** tab.
- Later browse for the downloaded spacex file.



- Once done select the schema andload the file.



In [1]:
```
!pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
```

```
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secret
d, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secret
e int.from_bytes instead
  from cryptography.utils import int_from_bytes
Collecting sqlalchemy==1.3.9
  Downloading SQLAlchemy-1.3.9.tar.gz (6.0 MB)
     |████████████████████████████████| 6.0 MB 14.3 MB/s eta 0:00:01
Building wheels for collected packages: sqlalchemy
  Stored in directory: /tmp/wsuser/.cache/pip/wheels/8c/76/0b/eb9eb3da7e2335e3577e3f96a0ae9f74f206e26457bd1a2bc8
Successfully built prettytable
Installing collected packages: sqlparse, prettytable, ipython-sql
Successfully installed ipython-sql-0.4.0 prettytable-0.7.2 sqlparse-0.4.2
```

Connect to the database

Let us first load the SQL extension and establish a connection with the database

```
In [2]: %load_ext sql
```

DB2 magic in case of old UI service credentials.

In the next cell enter your db2 connection string. Recall you created Service Credentials for your Db2 instance before. From the **uri** field of your Db2 service credentials copy everything after db2:// (except the double quote at the end) and paste it in the cell below after ibm_db_sa://

in the following format

in the following format

%sql ibm_db_sa://my-username:my-password\@my-hostname:my-port/my-db-name

DB2 magic in case of new UI service credentials.

- Use the following format.
- Add security=SSL at the end

%sql ibm_db_sa://my-username:my-password\@my-hostname:my-port/my-db-name?security=SSL

```
In [4]: %sql ibm_db_sa://cky43798:B3o4ThmQsD6Cuzq9@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb?security=SSL
```

Tasks

Now write and execute SQL queries to solve the assignment tasks.

Task 1

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

In [17]:
```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL
```

```
 * ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

Out[17]:

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

Task 2

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

In [18]:
```
%%sql
SELECT * FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[18]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Task 3

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

In [19]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)'
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-

7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done

Out[19]:

total_payload_mass_kg
45596

Task 4

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

In [20]:

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg
FROM SPACEXTBL
WHERE BOOSTER_VERSION = 'F9 v1.1'
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.

Out[20]:

avg_payload_mass_kg
2928

Task 5

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

Hint:Use min function

In [21]:

```
%%sql
SELECT MIN(DATE) AS first_successful_landing_date
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[21]:

first_successful_landing_date
2015-12-22

Task 6

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

In [22]:
```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (drone ship)'
    AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[22]:

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

Task 7

List the total number of successful and failure mission outcomes

In [23]:
```
%%sql
SELECT MISSION_OUTCOME, COUNT(*) AS total_number
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[23]:

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

Task 8

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

In [7]:
```
%%sql
SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[7]:

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

Task 9

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

In [9]:

```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = '2015'
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[9]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Task 10

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

In [26]:

```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS total_number
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY total_number DESC
```

* ibm_db_sa://cky43798:***@54a2f15b-5c0f-46df-8954-

7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb

Done.

Out[26]:

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

Reference Links

- Hands-on Lab : String Patterns, Sorting and Grouping - ref: SQL notes
- Hands-on Lab: Built-in functions – ref: SQL notes
- Hands-on Lab : Sub-queries and Nested SELECT Statements - ref: SQL notes
- Hands-on Tutorial: Accessing Databases with SQL magic - ref: SQL notes
- Hands-on Lab: Analyzing a real World Data Set- ref: SQL notes

https://github.com/brendensong/IBM-Data-Science-Professional-

Graded Quiz: Exploratory Analysis using SQL

Bookmarked

Quiz due Apr 15, 2022 06:08 +08 Completed

Question 1

1/1 point (graded)

Which of the following will retrieve the most recent date from the Spacex table?

- ○ SELECT MAXIMUM(Date) from SPACEXTBL
- ○ SELECT HIGHEST(Date) from SPACEXTBL
- ○ SELECT DATE FROM SPACEXTBL WHERE DATE=MAX(DATE)
- ◉ SELECT max(Date) from SPACEXTBL

✔

Submit You have used 1 of 2 attempts

Question 2

1/1 point (graded)

Which of the following queries display the minimum payload mass?

- ○
```
select payload_mass__kg_ from SPACEXTBL
group by booster_version
order by payload_mass__kg_ LIMIT 1
```
- ○ select payload_mass__kg_ from SPACEXTBL where payload_mass__kg_=(select min(payload_mass__kg_) from SPACEXTBL) LIMIT 1
- ○ select payload_mass__kg_ from SPACEXTBL order by payload_mass__kg_ LIMIT 1
- ◉ select min(payload_mass__kg_) from SPACEXTBL

✔

Submit You have used 1 of 2 attempts

Question 3

1/1 point (graded)

You are writing a query that will give you the averagel payload_mass_kg carried by the booster versions. The mass should be stored in the mass column. You want the result column to be called “Average_Payload_Mass”. Which of the following SQL queries is correct?

- SELECT count(PAYLOAD_MASS__KG_) as Average_Payload_Mass from SPACEXTBL
- SELECT avg(PAYLOAD_MASS__KG_) from SPACEXTBL
- SELECT avg(PAYLOAD_MASS__KG_) as Average_Payload_Mass from SPACEXTBL ✔

Submit You have used 2 of 2 attempts Show answer

Question 4

1/1 point (graded)

Which of the following query to display 10 records launched on Thursday?

- SELECT * FROM SPACEXTBL where DAYNAME(DATE)='Thursday' HAVING count(*)=10
- SELECT * FROM SPACEXTBL where DAYNAME(DATE)='Thursday' LIMIT 10 ✔

Submit You have used 1 of 1 attempt

Question 5

1/1 point (graded)

Which of the following are unique names launch sites mentioned in the spacex table?

- CCAFS LC-40,KSC LC-39A ✔
- CCAFS LC-40,KSC LC-39B
- CCAS LC-40,KSC LC-39
- None of the Above

Submit You have used 1 of 2 attempts Save

Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/5.%20Complete%20the%20EDA%20with%20Visualization%20lab.ipynb

Complete EDA with Visualization Lab

SpaceX Falcon 9 First Stage Landing Prediction

Assignment: Exploring and Preparing Data

Estimated time needed: **70** minutes

In this assignment, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is due to the fact that SpaceX can reuse the first stage.

In this lab, you will perform Exploratory Data Analysis and Feature Engineering.

Falcon 9 first stage will land successfully

Several examples of an unsuccessful landing are shown here:

Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.

Objectives

Perform exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib

- Exploratory Data Analysis
- Preparing Data Feature Engineering

Import Libraries and Define Auxiliary Functions

We will import the following libraries the lab

In [1]:
```
# andas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use this in our plotter function to plot da
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statisti
import seaborn as sns
```

Exploratory Data Analysis

First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

In [2]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

# If you were unable to complete the previous lab correctly you can uncomment and load this csv

# df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_par

df.head(5)
```

Out[2]:

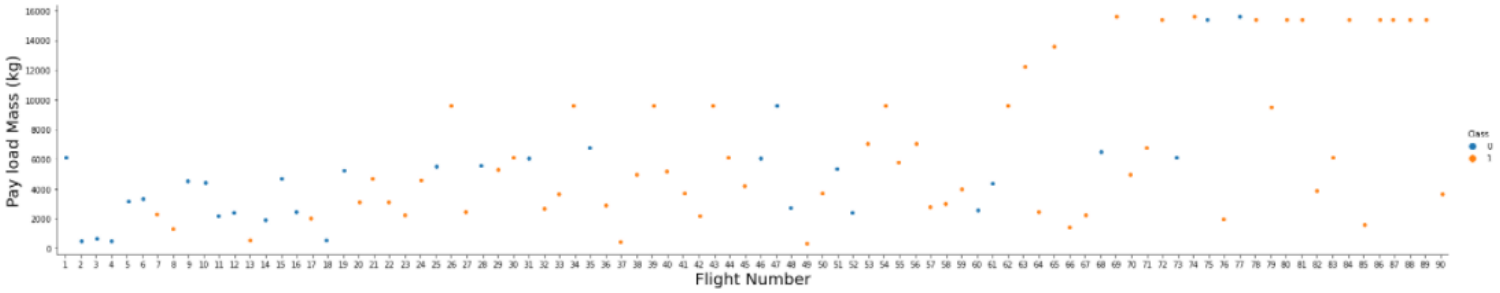
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False

Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
False	NaN	1.0	0	B0003	-80.577366	28.561857	0
False	NaN	1.0	0	B0005	-80.577366	28.561857	0
False	NaN	1.0	0	B0007	-80.577366	28.561857	0
False	NaN	1.0	0	B1003	-120.610829	34.632093	0
False	NaN	1.0	0	B1004	-80.577366	28.561857	0

First, let's try to see how the FlightNumber (indicating the continuous launch attempts.) and Payload variables would affect the launch outcome.

We can plot out the FlightNumber vs. PayloadMassand overlay the outcome of the launch. We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

In [3]:
```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



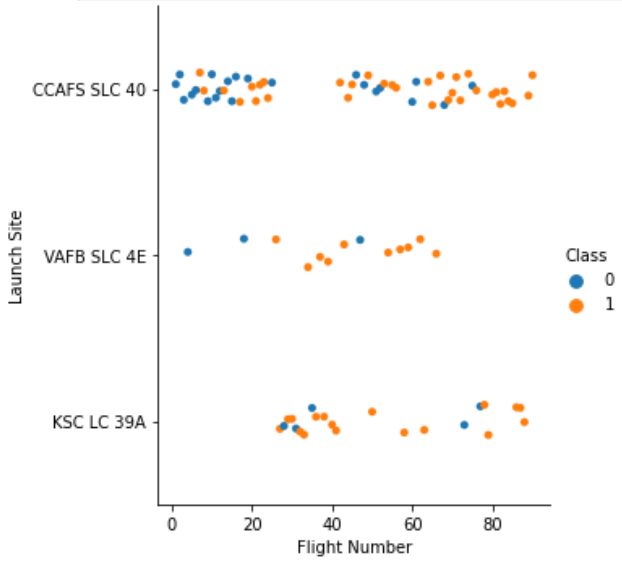
We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

Next, let's drill down to each site visualize its detailed launch records.

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function catplot to plot FlightNumber vs LaunchSite, set the parameter x parameter to FlightNumber,set the y to Launch Site and set the parameter hue to 'class'

In [4]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y='LaunchSite', x='FlightNumber', hue='Class', data=df)
plt.xlabel('Flight Number')
plt.ylabel('Launch Site')
plt.show()
```

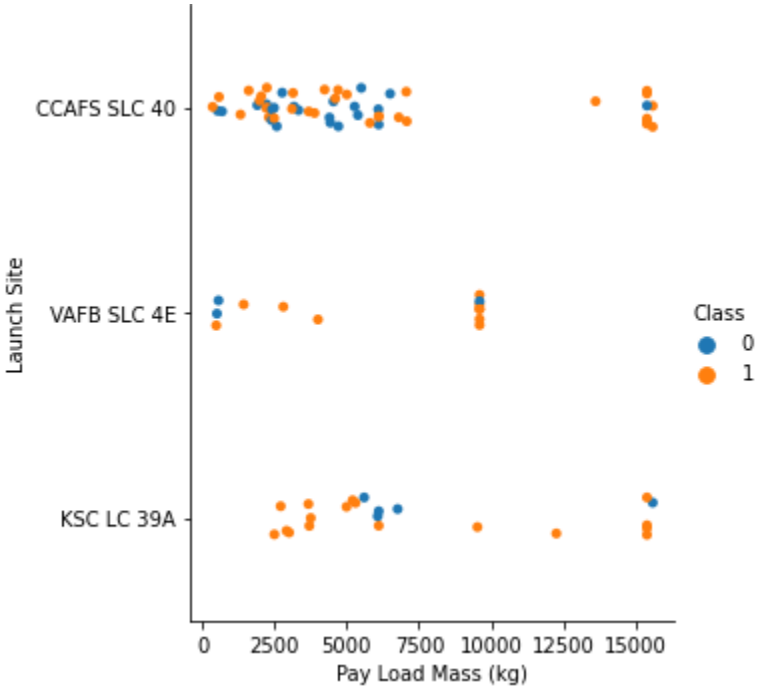


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

In [5]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class value
sns.catplot(y='LaunchSite', x='PayloadMass', hue='Class', data=df)
plt.xlabel('Pay Load Mass (kg)')
plt.ylabel('Launch Site')
plt.show()
```



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

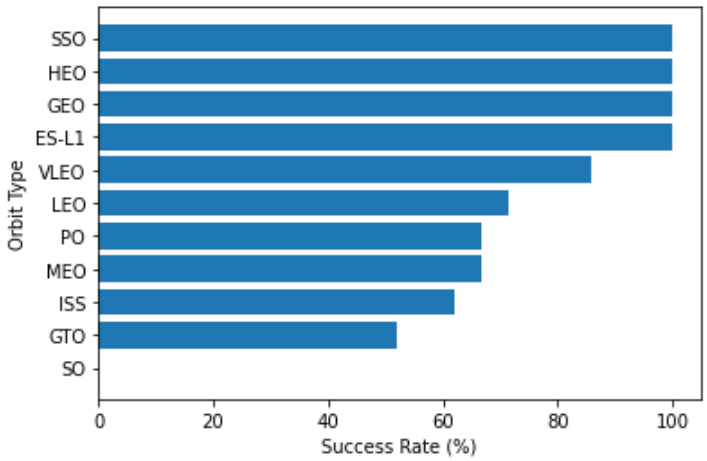
TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a bar chart for the success rate of each orbit

In [57]:

```
# HINT use groupby method on Orbit column and get the mean of Class column
df_sorted = df.groupby('Orbit').mean()['Class'].reset_index().sort_values(['Class'], ascending=True)

fig, ax = plt.subplots()

ax.barh(df_sorted.Orbit, df_sorted.Class * 100)

plt.xlabel('Success Rate (%)')
plt.ylabel('Orbit Type')
plt.show()
```



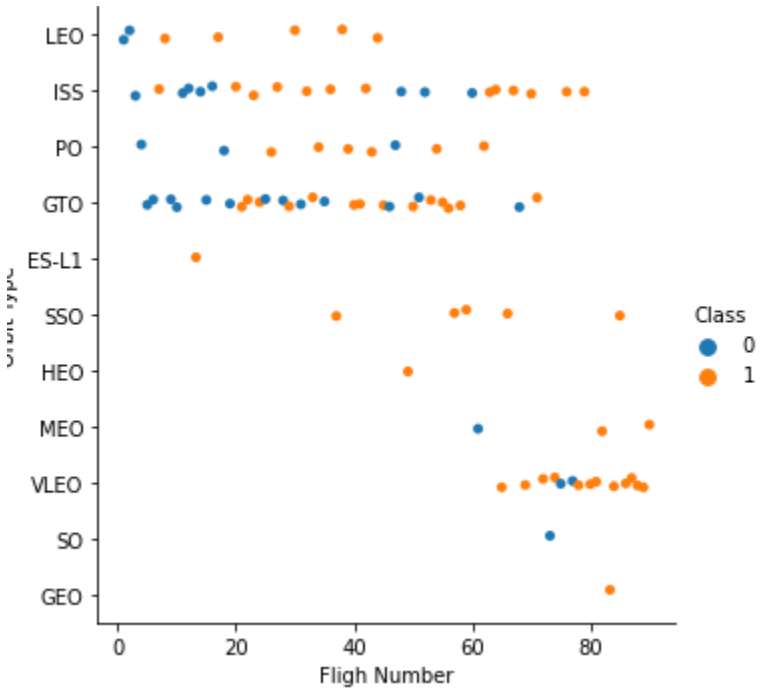
Analyze the ploted bar chart try to find which orbits have high sucess rate.

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [7]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='FlightNumber', y='Orbit', hue='Class', data=df)
plt.xlabel('Fligh Number')
plt.ylabel('Orbit Type')
plt.show()
```

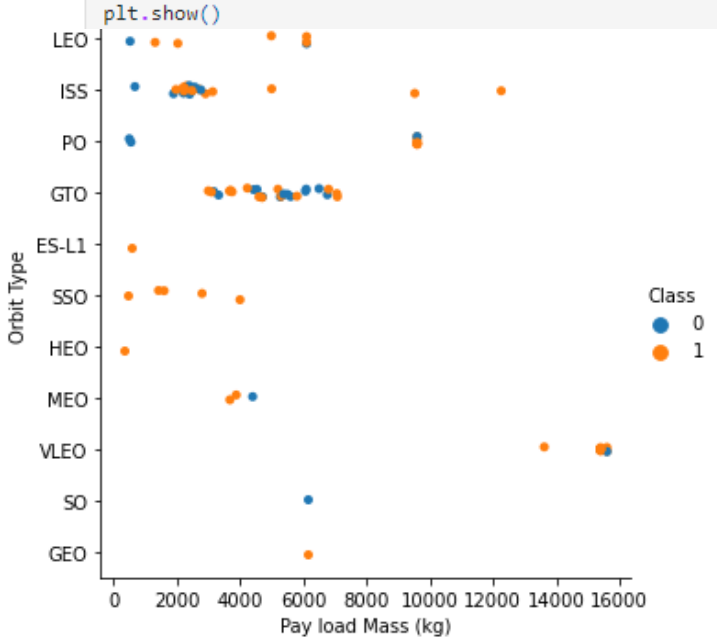


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [8]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x='PayloadMass', y='Orbit', hue='Class', data=df)
plt.xlabel('Pay load Mass (kg)')
plt.ylabel('Orbit Type')
plt.show()
```



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

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

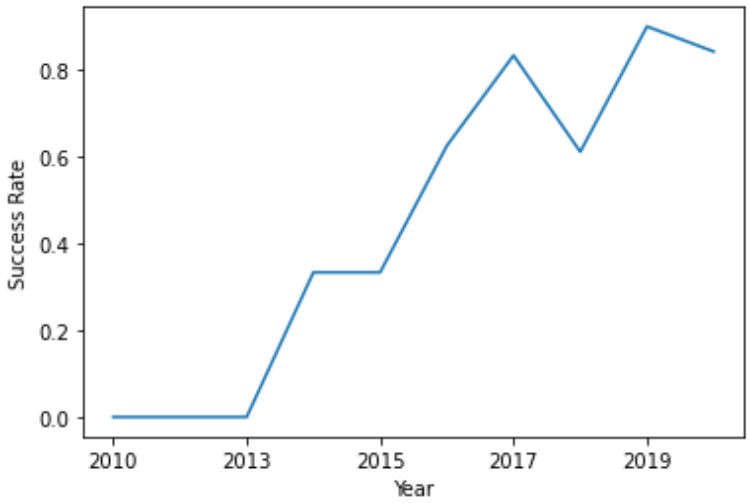
TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be Year and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

In [9]:
```
# A function to Extract years from the date
year=[]
def Extract_year(date):
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
```

In [10]:
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df.groupby(Extract_year(df['Date'])).mean()['Class'].plot(kind='line')
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

In [11]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Se
features.head()
```

Out[11]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

TASK 7: Create dummy variables to categorical columns

Use the function get_dummies and features dataframe to apply OneHotEncoder to the column Orbits, LaunchSite, LandingPad, and Serial. Assign the value to the variable features_one_hot, display the results using the method head. Your result dataframe must include all features including the encoded ones.

In [12]:
```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = pd.get_dummies(features, columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

Out[12]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0

5 rows × 80 columns

Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

TASK 8: Cast all numeric columns to float64

Now that our features_one_hot dataframe only contains numbers cast the entire dataframe to variable type float64

In [13]:

```python
# HINT: use astype function
features_one_hot.astype('float64')
```

Out[13]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
85	86.0	15400.000000	2.0	1.0	1.0	1.0	5.0	2.0	0.0
86	87.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0
87	88.0	15400.000000	6.0	1.0	1.0	1.0	5.0	5.0	0.0
88	89.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0
89	90.0	3681.000000	1.0	1.0	0.0	1.0	5.0	0.0	0.0

90 rows × 80 columns

Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

We can now export it to a **CSV** for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

features_one_hot.to_csv('dataset_part_3.csv', index=False)

Authors

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Nayef Abou Tayoun is a Data Scientist at IBM and pursuing a Master of Management in Artificial intelligence degree at Queen's Univ.

Graded Quiz: Exploratory Data Analysis

Bookmarked

Quiz due Apr 15, 2022 06:08 +08 Completed

Question 1

1/1 point (graded)

What type of data does a Bar Chart best represent?

- ○ Location Data
- ○ Numerical
- ◉ Categorical
- ○ None of the above

✔

Submit You have used 1 of 2 attempts

Question 2

1/1 point (graded)

What are the total number of columns in the features dataframe after applying one hot encoding to columns Orbits, LaunchSite, LandingPad and Serial . Here the features dataframe consists of the following columns FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial'

- ○ 120
- ◉ 80
- ○ 83
- ○ 96

✔

Submit You have used 1 of 2 attempts Save Show ansv

Question 3

1/1 point (graded)

The catplot code to show the scatterplot of FlightNumber vs LaunchSite with x as FlightNumber, and y to Launch Site and hue to 'Class' is sns.catplot(y="LaunchSite",x="FlightNumber",hue="Class", data=df, aspect = 1) plt.ylabel("Launch Site",fontsize=15) plt.xlabel("Flight Number",fontsize=15) plt.show()

- ◉ True
- ○ False

✔

Module Introduction and Learning Objectives

Module Introduction

After completing this interactive visual analytics module, you will be able to:

Learning Objectives

- Build an interactive map to analyze the launch site proximity with Folium
- Build a dashboard to analyze launch records interactively with Plotly Dash

Data Visualization and Dashboard Overview

Welcome to the Interactive Visual Analytics and Dashboard module. You will use this to build a Dashboard for stakeholders.
Interactive visual analytics enables users to explore and manipulate data in an interactive
and real-time way.
Common interactions including zoom-in and zoom-out, pan, filter, search, and link.
With interactive visual analytics, users could find visual patterns faster and more effectively. Instead of presenting your findings in static graphs, interactive data visualization, or dashboarding, can always tell a more appealing story.

In this module, you will be using Folium and Plotly Dash to build an interactive map and dashboard to perform interactive visual analytics. The first part of this module will be focused on analyzing launch site geo and proximities with Folium. We will first mark the launch site locations and their close proximities on an interactive map. Then, we can explore the map with those markers and try to discover any patterns from them.

Finally, we should be able to explain how to choose an optimal launch site.
Next, you will be building a dashboard application with the Python Plotly Dash package.
This dashboard application contains input components such as a dropdown list and a range
slider to interact with a pie chart and a scatter point chart.
You will be guided to build this dashboard application in an instructional lab.
After the dashboard is built, you can use it to find more insights from the SpaceX dataset
more easily than with static graphs.

Interactive Visual Analytics

- Enable direct data exploration and analytics
- Zoom-in/out, Pan, Filter, Search, Link, etc.
- Identify patterns faster and more effectively
- More appealing stories

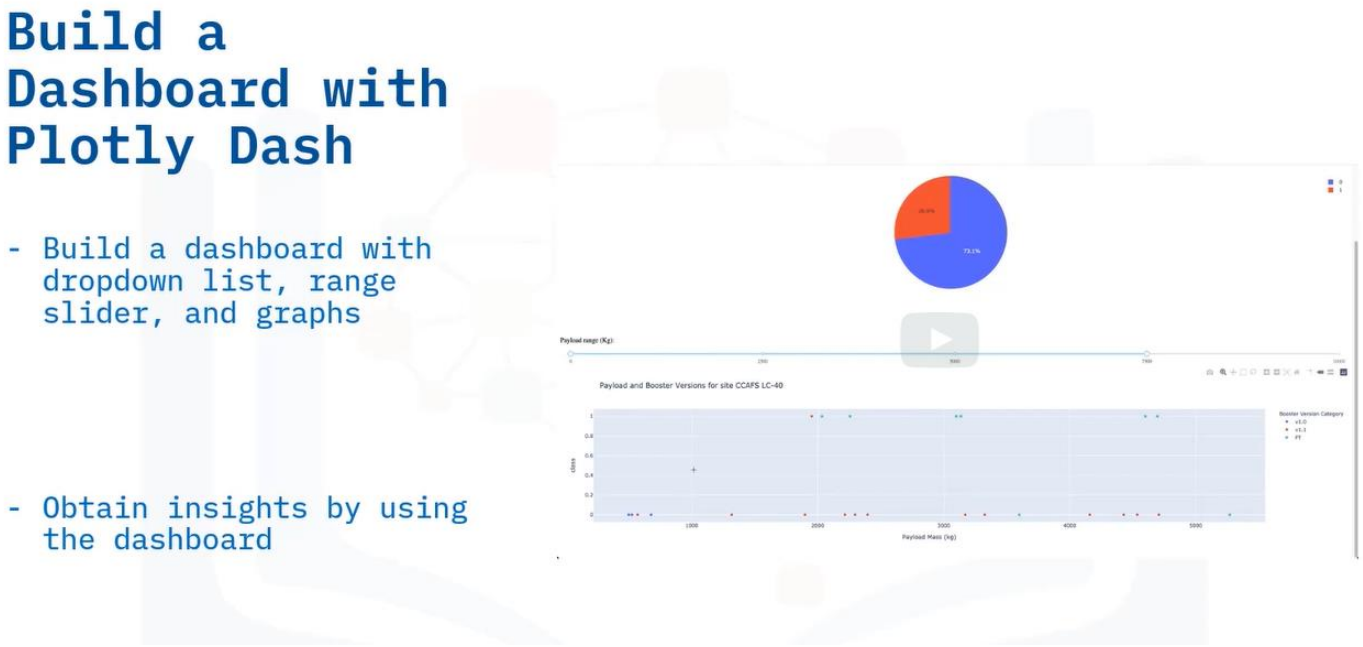
Folium Plotly

Interactive visual analytics enables users to explore and manipulate data in an interactive and real-time way. Common interactions including zoom-in and zoom-out, pan, filter, search, and link. With interactive visual analytics, users could find visual patterns faster and more effectively. Instead of presenting your findings in static graphs, interactive data visualization, or dashboarding (not static graphs), can always tell a more appealing story.

Analyze Launch Site Geo Data with Folium

- Mark the locations and proximities of launch sites
- Discover patterns via exploring the map
- Explain how to choose an optimal launch site locations

The first part of this module will be focused on analyzing launch site geo and proximities with Folium.
We will first mark the launch site locations and their close proximities on an interactive map.
Then, we can explore the map with those markers and try to discover any patterns from them.



Finally, we should be able to explain how to choose an optimal launch site. Next, you will be building a dashboard application with the Python Plotly Dash package. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. You will be guided to build this dashboard application in an instructional lab. After the dashboard is built, you can use it to find more insights from the SpaceX dataset more easily than with static graphs.

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/wiki/10.3.Interactive-Visual-Analytics-and-Dashboard

10.3.Interactive Visual Analytics and Dashboard

Seongjoo Brenden Song edited this page on Nov 7, 2021 · 4 revisions

In this module, you will build a dashboard to analyze launch records interactively with Plotly Dash. You will then build an interactive map to analyze the launch site proximity with Folium.

Learning Objectives

- Build a dashboard to analyze launch records interactively with Plotly Dash.
- Build an interactive map to analyze the launch site proximity with Folium.

- Complete the Data Visualization with Folium (Jupyter Notebook)
- Complete the Data Visualization with Folium (IBM Cloud URL for interactive maps)
- Build an Interactive Dashboard with Ploty Dash (code)
- Build an Interactive Dashboard with Ploty Dash (dashboard URL)

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/6.%20Complete%20the%20Interactive%20Visual%20Analytics%20with%20.ipynb

Complete the data Visualization with Folium (Jupyter Notebook)

Launch Sites Locations Analysis with Folium

Estimated time needed: **40** minutes

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal 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.

In the previous exploratory data analysis labs, you have visualized the SpaceX launch dataset using matplotlib and seaborn and discovered some preliminary correlations between the launch site and success rates. In this lab, you will be performing more interactive visual analytics using Folium.

Objectives

This lab contains the following tasks:

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

After completed the above tasks, you should be able to find some geographical patterns about launch sites.

Let's first import required Python packages for this lab:

In [1]:
```
!pip3 install folium
!pip3 install wget
```

opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/site-packages/secretstorage/util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprecated, use int.from_bytes instead

Successfully built wget

Installing collected packages: wget

Successfully installed wget-3.2

In [2]:
```
import folium
import wget
import pandas as pd
```

In [3]:
```
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

If you need to refresh your memory about folium, you may download and refer to this previous folium lab:

Generating Maps with Python

Task 1: Mark all launch sites on a map

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

The following dataset with the name spacex_launch_geo.csv is an augmented dataset with latitude and longitude added for each site.

In [4]:
```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_g
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

In [5]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df
```

Out[5]:

	Launch Site	Lat	Long	class
0	CCAFS LC-40	28.562302	-80.577356	0
1	CCAFS SLC-40	28.563197	-80.576820	1
2	KSC LC-39A	28.573255	-80.646895	1
3	VAFB SLC-4E	34.632834	-120.610746	0

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

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

In [6]:
```
# Start location is NASA Johnson Space Center
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. For example,

In [7]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#007fd3', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#007fd3;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Out[7]: Make this Notebook Trusted to load map: File -> Trust Notebook

and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

Now, let's add a circle for each launch site in data frame `launch_sites`

TODO: Create and add `folium.Circle` and `folium.Marker` for each launch site on the site map

An example of folium.Circle:

`folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(...))`

An example of folium.Marker:

`folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'label', ))`

In [93]:
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
for lat, lng, label in zip(launch_sites_df['Lat'], launch_sites_df['Long'], launch_sites_df['Launch Site']):
    coordinate = [lat, lng]
    circle = folium.Circle(coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup(label))
    marker = folium.map.Marker(
        coordinate,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % label,
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```

Out[93]: Make this Notebook Trusted to load map: File -> Trust Notebook

The generated map with marked launch sites should look similar to the following:

Now, you can explore the map by zoom-in/out the marked areas , and try to answer the following questions:

- Are all launch sites in proximity to the Equator line?
- Are all launch sites in very close proximity to the coast?

Also please try to explain your findings.

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

Next, let's try to enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Recall that data frame spacex_df has detailed launch records, and the class column indicates if this launch was successful or not

In [78]:
```
spacex_df.tail(10)
```

Out[78]:

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

Next, 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)

Note that a launch only happens in one of the four launch sites, which means many launch records will have the

exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

Let's first create a MarkerCluster object

In [94]:
```
marker_cluster = MarkerCluster()
```

TODO: Create a new column in launch_sites dataframe called marker_color to store the marker colors based on the class value

In [80]:
```
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
def assign_marker_color_LSDF(launch_class):
    if launch_class == 1:
        return 'green'
    else:
        return 'red'

launch_sites_df['marker_color'] = launch_sites_df['class'].apply(assign_marker_color_LSDF)
launch_sites_df
```

Out[80]:

	Launch Site	Lat	Long	class	marker_color
0	CCAFS LC-40	28.562302	-80.577356	0	red
1	CCAFS SLC-40	28.563197	-80.576820	1	green
2	KSC LC-39A	28.573255	-80.646895	1	green
3	VAFB SLC-4E	34.632834	-120.610746	0	red

In [81]:

```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```

Out[81]:

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

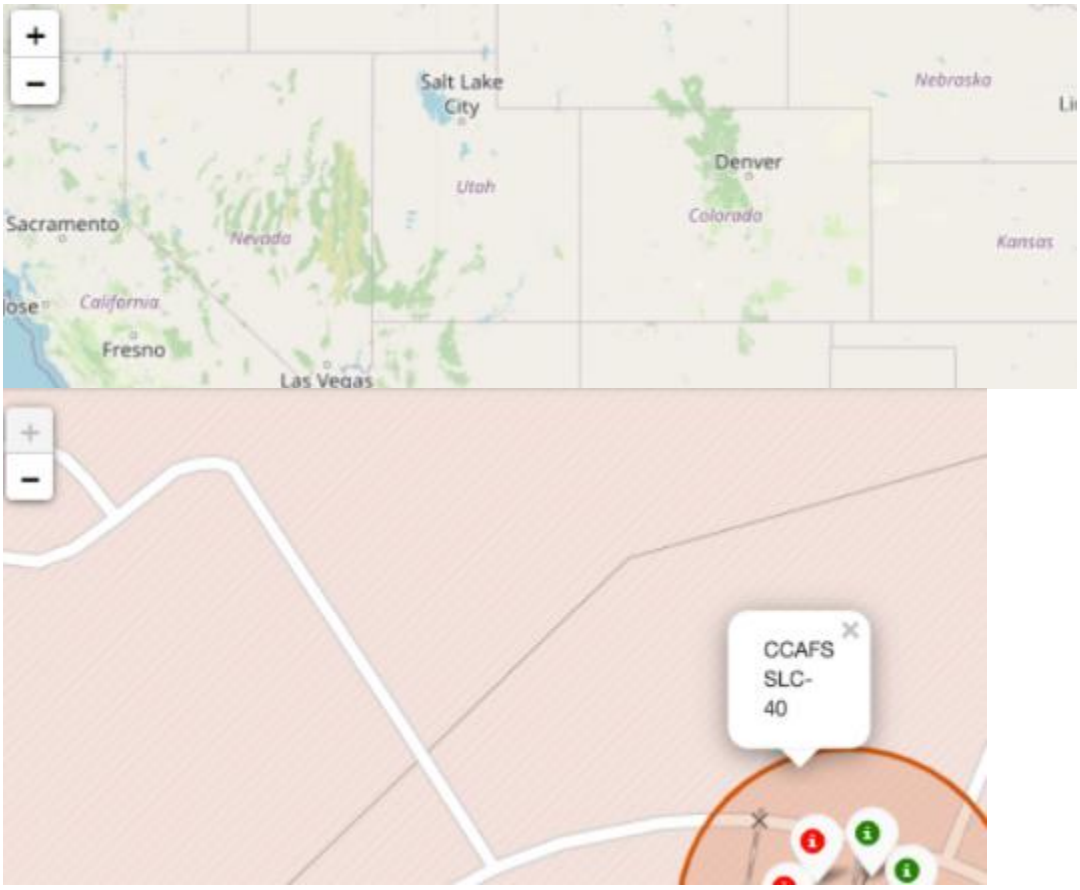
TODO: For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster

In [95]:

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for lat, lng, label, color in zip(spacex_df['Lat'], spacex_df['Long'], spacex_df['Launch Site'], spacex_df['marker_color']):
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    coordinate = [lat, lng]
    marker = folium.Marker(
        coordinate,
        icon=folium.Icon(color='white', icon_color=color),
        popup=label
    )
    marker_cluster.add_child(marker)

site_map
```

Out[95]: Make this Notebook Trusted to load map: File -> Trust Notebook

Your updated map may look like the following screenshots:



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

TASK 3: Calculate the distances between a launch site to its proximities

Next, we need to explore and analyze the proximities of launch sites.

Let's first add a MousePosition on the map to get coordinate for a mouse over a point on the map. As such, while you are exploring the map, you can easily find the coordinates of any points of interests (such as railway)

In [96]:
```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```

Out[96]: Make this Notebook Trusted to load map: File -> Trust Notebook

Now zoom in to a launch site and explore its proximity to see if you can easily find any railway, highway, coastline, etc. Move your mouse to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

You can calculate the distance between two points on the map based on their Lat and Long values using the following method:

In [97]:
```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

TODO: Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

In [98]:
```
# find coordinate of the closet coastline
coastline_lat = 28.56398
coastline_lon = -80.56809
launch_site_lat = 28.56321
launch_site_lon = -80.57683
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
```

TODO: After obtained its coordinate, create a folium.Marker to show the distance

In [99]:
```
# Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon property
coast_coordinates = [coastline_lat, coastline_lon]
distance_marker = folium.Marker(
    coast_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_coastline),
        )
    )
distance_marker.add_to(site_map)
site_map
```

Out[99]: Make this Notebook Trusted to load map: File -> Trust Notebook

TODO: Draw a `PolyLine` between a launch site to the selected coastline point

In [100…

```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[coast_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
```

Out[100…

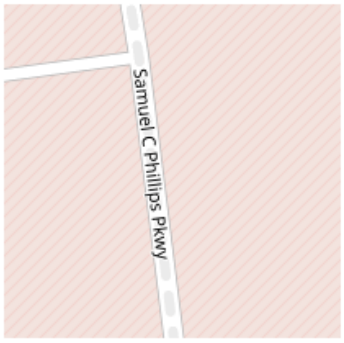
Make this Notebook Trusted to load map: File -> Trust Notebook

Your updated map with distance line should look like the following screenshot:

TODO: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use MousePosition to find the their coordinates on the map first

A railway map symbol may look like this:

A highway map symbol may look like this:



A city map symbol may look like this:

In [101…

```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the closest city(Titusville) to the launch site
city_lat = 28.61208
city_lon = -80.80764
distance_city = calculate_distance(launch_site_lat, launch_site_lon, city_lat, city_lon)

city_coordinates = [city_lat, city_lon]
distance_marker = folium.Marker(
    city_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[city_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[101…

Make this Notebook Trusted to load map: File -> Trust Notebook

In [102…

```
# Draw a line between the closest railway to the launch site
railway_lat = 28.57208
railway_lon = -80.58527
distance_railway = calculate_distance(launch_site_lat, launch_site_lon, railway_lat, railway_lon)

railway_coordinates = [railway_lat, railway_lon]
distance_marker = folium.Marker(
    railway_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_railway),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[railway_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[102…

Make this Notebook Trusted to load map: File -> Trust Notebook

In [103…

```
# Draw a line between the closest highway to the launch site
highway_lat = 28.56478
highway_lon = -80.57103
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, highway_lat, highway_lon)

highway_coordinates = [highway_lat, highway_lon]
distance_marker = folium.Marker(
    highway_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_highway),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[highway_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[103… Make this Notebook Trusted to load map: File -> Trust Notebook

In [104…

```
# Draw a line between the closest city(Cape Canaveral) to the launch site
city_2_lat = 28.40159
city_2_lon = -80.6042
distance_city_2 = calculate_distance(launch_site_lat, launch_site_lon, city_2_lat, city_2_lon)

city_2_coordinates = [city_2_lat, city_2_lon]
distance_marker = folium.Marker(
    city_2_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city_2),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[city_2_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[104… Make this Notebook Trusted to load map: File -> Trust Notebook

In [105…

```
# Draw a line between the closest city(Lompoc) to the launch site(Space Launch Complex 4)
launch_site_4_lat = 34.63286
launch_site_4_lon = -120.61074
launch_site_4_coordinates = [launch_site_4_lat, launch_site_4_lon]

city_Lompoc_lat = 34.63879
city_Lompoc_lon = -120.45788
distance_city_Lompoc = calculate_distance(launch_site_4_lat, launch_site_4_lon, city_Lompoc_lat, city_Lompoc_lon)

city_Lompoc_coordinates = [city_Lompoc_lat, city_Lompoc_lon]

distance_marker = folium.Marker(
    city_Lompoc_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city_Lompoc),
        )
    )
distance_marker.add_to(site_map)

lines=folium.PolyLine(locations=[city_Lompoc_coordinates, launch_site_4_coordinates], weight=1)
site_map.add_child(lines)
```

```
# Draw a line between the closest railway to the launch site(Space Launch Complex 4)
railway_4_lat = 34.63677
railway_4_lon = -120.6236
distance_railway_4 = calculate_distance(launch_site_4_lat, launch_site_4_lon, railway_4_lat, railway_4_lon)

railway_4_coordinates = [railway_4_lat, railway_4_lon]

distance_marker = folium.Marker(
    railway_4_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_railway_4),
        )
    )
distance_marker.add_to(site_map)

lines=folium.PolyLine(locations=[railway_4_coordinates, launch_site_4_coordinates], weight=1)
site_map.add_child(lines)

site_map
```

Out[105…

Make this Notebook Trusted to load map: File -> Trust Notebook

After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Also please try to explain your findings.

Next Steps:

Now you have discovered many interesting insights related to the launch sites' location using folium, in a very interactive way. Next, you will need to build a dashboard using Ploty Dash on detailed launch records.

Authors

Yan Luo

https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/9efccadc-2cdd-435e-9339-b5770ac84a65/view?access_token=42452a1d4ebb5187f1b2f20fb6438c4824ea6258ac211029bb5b349dc4df1ae0

Complete Data Visualization with Folium (IBM Cloud URL fro Interactive Maps

Launch Sites Locations Analysis with Folium¶

Estimated time needed: **40** minutes

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal 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.

In the previous exploratory data analysis labs, you have visualized the SpaceX launch dataset using matplotlib and seaborn and discovered some preliminary correlations between the launch site and success rates. In this lab, you will be performing more interactive visual analytics using Folium.

Objectives

This lab contains the following tasks:

- **TASK 1:** Mark all launch sites on a map
- **TASK 2:** Mark the success/failed launches for each site on the map
- **TASK 3:** Calculate the distances between a launch site to its proximities

After completed the above tasks, you should be able to find some geographical patterns about launch sites.

Let's first import required Python packages for this lab:

```
In [1]: !pip3 install folium
        !pip3 install wget
```

```
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/sit
_from_bytes is deprecated, use int.from_bytes inste
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-OpenCE/lib/python3.7/sit
m_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Collecting folium
  Downloading folium-0.12.1-py2.py3-none-any.whl (9
     |████████████████████████████████| 94 kB 5.4 M
Requirement already satisfied: jinja2>=2.9 in /opt/
1.3)
Collecting branca>=0.3.0
  Downloading branca-0.4.2-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy in /opt/conda/
```

In [2]:
```
import folium
import wget
import pandas as pd
```

In [3]:
```
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

If you need to refresh your memory about folium, you may download and refer to this previous folium lab:

Generating Maps with Python

Task 1: Mark all launch sites on a map

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

The following dataset with the name spacex_launch_geo.csv is an augmented dataset with latitude and longitude added for each site

In [4]:
```
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_launch_geo.csv')
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

In [5]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df
```

Out[5]:

	Launch Site	Lat	Long	class
0	CCAFS LC-40	28.562302	-80.577356	0
1	CCAFS SLC-40	28.563197	-80.576820	1
2	KSC LC-39A	28.573255	-80.646895	1
3	VAFB SLC-4E	34.632834	-120.610746	0

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

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

In [6]:
```
# Start location is NASA Johnson Space Center
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. For example,

In [7]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#007fd3', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#007fd3;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Out[7]:

and you should find a small yellow circle near the city of Houston and you can zoom-in to see a larger circle.

Now, let's add a circle for each launch site in data frame launch_sites

TODO: Create and add folium.Circle and folium.Marker for each launch site on the site map

An example of folium.Circle:

folium.Circle(coordinate, radius=1000, color='#000000', fill=True).add_child(folium.Popup(...))

An example of folium.Marker:

folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html=' %s ' % 'label',))

In [93]:
```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a p
opup label
for lat, lng, label in zip(launch_sites_df['Lat'], launch_sites_df['Long'], launch_sites_df['Launch Site']):
    coordinate = [lat, lng]
    circle = folium.Circle(coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup(label))
    marker = folium.map.Marker(
        coordinate,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % label,
            )
        )
    site_map.add_child(circle)
    site_map.add_child(marker)

site_map
```

Out[93]:

The generated map with marked launch sites should look similar to the following:

Now, you can explore the map by zoom-in/out the marked areas , and try to answer the following questions:

Are all launch sites in proximity to the Equator line?
Are all launch sites in very close proximity to the coast?
Also please try to explain your findings.

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

Next, let's try to enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Recall that data frame spacex_df has detailed launch records, and the class column indicates if this launch was successful or not

```
In [78]: spacex_df.tail(10)
```

Out[78]:

	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

Next, 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)

Note that a launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. Marker clusters can be a good way to simplify a map containing many markers having the same coordinate.

Let's first create a MarkerCluster object

```
In [94]: marker_cluster = MarkerCluster()
```

TODO: Create a new column in launch_sites dataframe called marker_color to store the marker colors based on the class value

In [80]:
```
# Apply a function to check the value of `class` column
# If class=1, marker_color value will be green
# If class=0, marker_color value will be red
def assign_marker_color_LSDF(launch_class):
    if launch_class == 1:
        return 'green'
    else:
        return 'red'

launch_sites_df['marker_color'] = launch_sites_df['class'].apply(assign_marker_color_LSDF)
launch_sites_df
```

Out[80]:

	Launch Site	Lat	Long	class	marker_color
0	CCAFS LC-40	28.562302	-80.577356	0	red
1	CCAFS SLC-40	28.563197	-80.576820	1	green
2	KSC LC-39A	28.573255	-80.646895	1	green
3	VAFB SLC-4E	34.632834	-120.610746	0	red

In [81]:
```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```

Out[81]:

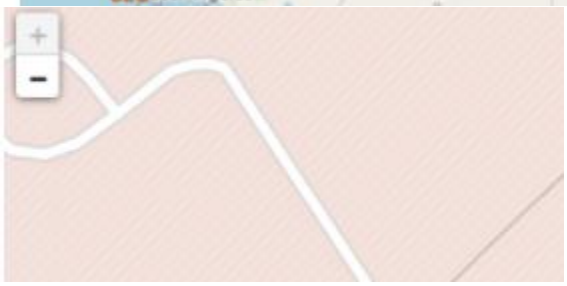
	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

TODO: For each launch result in spacex_df data frame, add a folium.Marker to marker_cluster

In [95]:
```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
for lat, lng, label, color in zip(spacex_df['Lat'], spacex_df['Long'], spacex_df['Launch Site'], spacex_df['marker_color']):
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    coordinate = [lat, lng]
    marker = folium.Marker(
        coordinate,
        icon=folium.Icon(color='white', icon_color=color),
        popup=label
    )
    marker_cluster.add_child(marker)

site_map
```

Out[95]:

Your updated map may look like the following screenshots:



From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

TASK 3: Calculate the distances between a launch site to its proximities

Next, we need to explore and analyze the proximities of launch sites.

Let's first add a MousePosition on the map to get coordinate for a mouse over a point on the map. As such, while you are exploring the map, you can easily find the coordinates of any points of interests (such as railway)

In [96]:
```
# Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```

Out[96]:

Now zoom in to a launch site and explore its proximity to see if you can easily find any railway, highway, coastline, etc. Move your mouse to these points and mark down their coordinates (shown on the top-left) in order to the distance to the launch site.

You can calculate the distance between two points on the map based on their Lat and Long values using the following method:

In [97]:
```
from math import sin, cos, sqrt, atan2, radians

def calculate_distance(lat1, lon1, lat2, lon2):
    # approximate radius of earth in km
    R = 6373.0

    lat1 = radians(lat1)
    lon1 = radians(lon1)
    lat2 = radians(lat2)
    lon2 = radians(lon2)

    dlon = lon2 - lon1
    dlat = lat2 - lat1

    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2
    c = 2 * atan2(sqrt(a), sqrt(1 - a))

    distance = R * c
    return distance
```

TODO: Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.

In [98]:
```
# find coordinate of the closet coastline
coastline_lat = 28.56398
coastline_lon = -80.56809
launch_site_lat = 28.56321
launch_site_lon = -80.57683
distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
```

TODO: After obtained its coordinate, create a folium.Marker to show the distance

In [99]:
```
# Create and add a folium.Marker on your selected closest coastline point on the map
# Display the distance between coastline point and launch site using the icon property
coast_coordinates = [coastline_lat, coastline_lon]
distance_marker = folium.Marker(
    coast_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_coastline),
        )
    )
distance_marker.add_to(site_map)
site_map
```

Out[99]:

TODO: Draw a PolyLine between a launch site to the selected coastline point

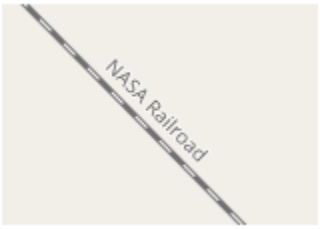
In [100]:
```
# Create a `folium.PolyLine` object using the coastline coordinates and launch site coordinate
launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[coast_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
```

Out[100]:

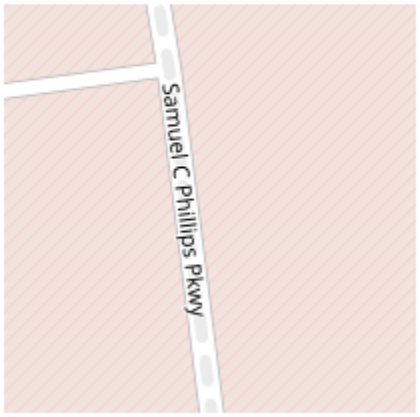
Your updated map with distance line should look like the following screenshot:

TODO: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use MousePosition to find the their coordinates on the map first

A railway map symbol may look like this:



A highway map symbol may look like this:



A city map symbol may look like this:

In [101]:
```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the closest city(Titusville) to the launch site
city_lat = 28.61208
city_lon = -80.80764
distance_city = calculate_distance(launch_site_lat, launch_site_lon, city_lat, city_lon)

city_coordinates = [city_lat, city_lon]
distance_marker = folium.Marker(
    city_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[city_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[101]:

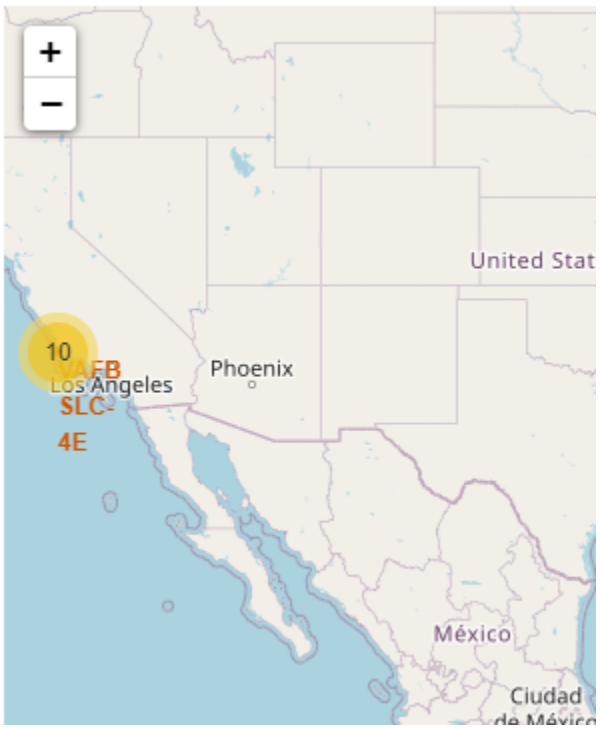
In [102]:
```
# Draw a line between the closest railway to the launch site
railway_lat = 28.57208
railway_lon = -80.58527
distance_railway = calculate_distance(launch_site_lat, launch_site_lon, railway_lat, railway_lon)

railway_coordinates = [railway_lat, railway_lon]
distance_marker = folium.Marker(
    railway_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_railway),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[railway_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[102]:

In [103]:

```
# Draw a line between the closest highway to the launch site
highway_lat = 28.56478
highway_lon = -80.57103
distance_highway = calculate_distance(launch_site_lat, launch_site_lon, highway_lat, highway_lon)

highway_coordinates = [highway_lat, highway_lon]
distance_marker = folium.Marker(
    highway_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_highway),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[highway_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[103]:



In [104]:
```
# Draw a line between the closest city(Cape Canaveral) to the launch site
city_2_lat = 28.40159
city_2_lon = -80.6042
distance_city_2 = calculate_distance(launch_site_lat, launch_site_lon, city_2_lat, city_2_lon)

city_2_coordinates = [city_2_lat, city_2_lon]
distance_marker = folium.Marker(
    city_2_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city_2),
        )
    )
distance_marker.add_to(site_map)

launch_site_coordinates = [launch_site_lat, launch_site_lon]
lines=folium.PolyLine(locations=[city_2_coordinates, launch_site_coordinates], weight=1)
site_map.add_child(lines)
site_map
```

Out[104]:

In [105]:
```
# Draw a line between the closest city(Lompoc) to the launch site(Space Launch Complex 4)
launch_site_4_lat = 34.63286
launch_site_4_lon = -120.61074
launch_site_4_coordinates = [launch_site_4_lat, launch_site_4_lon]

city_Lompoc_lat = 34.63879
city_Lompoc_lon = -120.45788
distance_city_Lompoc = calculate_distance(launch_site_4_lat, launch_site_4_lon, city_Lompoc_lat, city_Lompoc_lon)

city_Lompoc_coordinates = [city_Lompoc_lat, city_Lompoc_lon]

distance_marker = folium.Marker(
    city_Lompoc_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_city_Lompoc),
        )
    )
distance_marker.add_to(site_map)

lines=folium.PolyLine(locations=[city_Lompoc_coordinates, launch_site_4_coordinates], weight=1)
site_map.add_child(lines)
```

```
# Draw a line between the closest coast to the launch site(Space Launch Complex 4)
west_coast_lat = 34.63698
west_coast_lon = -120.6245
distance_west_coast = calculate_distance(launch_site_4_lat, launch_site_4_lon, west_coast_lat, west_coast_lon)

west_coast_coordinates = [west_coast_lat, west_coast_lon]

distance_marker = folium.Marker(
    west_coast_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_west_coast)
        )
    )
distance_marker.add_to(site_map)

lines=folium.PolyLine(locations=[west_coast_coordinates, launch_site_4_coordinates], weight=1)
site_map.add_child(lines)
```

```
# Draw a line between the closest railway to the launch site(Space Launch Complex 4)
railway_4_lat = 34.63677
railway_4_lon = -120.6236
distance_railway_4 = calculate_distance(launch_site_4_lat, launch_site_4_lon, railway_4_lat, railway_4_lon)

railway_4_coordinates = [railway_4_lat, railway_4_lon]

distance_marker = folium.Marker(
    railway_4_coordinates,
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_railway_4),
        )
    )
distance_marker.add_to(site_map)

lines=folium.PolyLine(locations=[railway_4_coordinates, launch_site_4_coordinates], weight=1)
site_map.add_child(lines)

site_map
```

Out[105]:

After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Also please try to explain your findings.

Next Steps:

Now you have discovered many interesting insights related to the launch sites' location using folium, in a very interactive way. Next, you will need to build a dashboard using Ploty Dash on detailed launch records.

Authors

Yan Luo

IBM-Data-Science-Professional-Certificate/10_Applied_Data_Science_Capstone/7. Build an Interactive Dashboard with Ploty
https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/7.%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.py

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

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# 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',
                                             options=[
                                                 {'label': 'All Sites', 'value': 'ALL'},
                                                 {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                                 {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                                 {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                                 {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'},
                                             ],
                                             value='ALL',
                                             placeholder="Select a Launch Site here",
                                             searchable=True
                                             ),
                                html.Br(),

```

```
                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                #dcc.RangeSlider(id='payload-slider',...)
                                dcc.RangeSlider(id='payload-slider',
                                                min=0, max=10000, step=1000,
                                                marks={0: '0', 2500: '2500', 5000: '5000', 7500: '7500', 10000: '10000'},
                                                value=[min_payload, max_payload]),
                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])

# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
# Function decorator to specify function input and output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filtered_df, values='class',
        names='Launch Site',
        title='Total Success Launches By Site')
        return fig
    else:
        # return the outcomes piechart for a selected site
        filtered_df = spacex_df[spacex_df['Launch Site'] == entered_site]
        filtered_df = filtered_df.groupby(['Launch Site', 'class']).size().reset_index(name='class count')
        fig = px.pie(filtered_df, values='class count',
        names='class',
        title=f'Total Success Launched for site {entered_site}')
        return fig
```

```python
# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              [Input(component_id='site-dropdown', component_property='value'),
              Input(component_id='payload-slider', component_property='value')])
def get_scatter_chart(entered_site, payload):
    low, high = payload
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] > low) & (spacex_df['Payload Mass (kg)'] < high)]
    if entered_site == 'ALL':
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class',
        color='Booster Version Category',
        title='Correlation between Payload and Success for all Sites')
        return fig
    else:
        fig = px.scatter(filtered_df[filtered_df['Launch Site'] == entered_site],
        x='Payload Mass (kg)', y='class',
        color='Booster Version Category',
        title=f'Correlation between Payload and Success for site {entered_site}')
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```

pythonanywhere

Coming Soon!

This is going to be another great website hosted by PythonAnywhere.

PythonAnywhere lets you host, run, and code Python in the cloud. Our free plan gives you access to machines with everything already set up for you. You can develop and host your website or any other code directly from your browser without having to install software or manage your own server.

Need more power? Upgraded plans start at $5/month.

You can find out more about PythonAnywhere here.

Developer info

Hi! If this is your PythonAnywhere-hosted site, then you're almost there — you just need to create a web app to handle this domain.

Go to the "Web" tab inside PythonAnywhere and click "Add a new web app". If you already have a web app and you want to use the same code for this domain (say because you've just upgraded and want the site you built at `yourusername.pythonanywhere.com` to run on `www.yourdomain.com`) then this help page should explain everything.

If you're having problems getting it all working, drop us a line at support@pythonanywhere.com, or in the forums, or using the "Send feedback" link on the site. We'll get back to you as fast as we can!

Module 4 Predictive Analysis

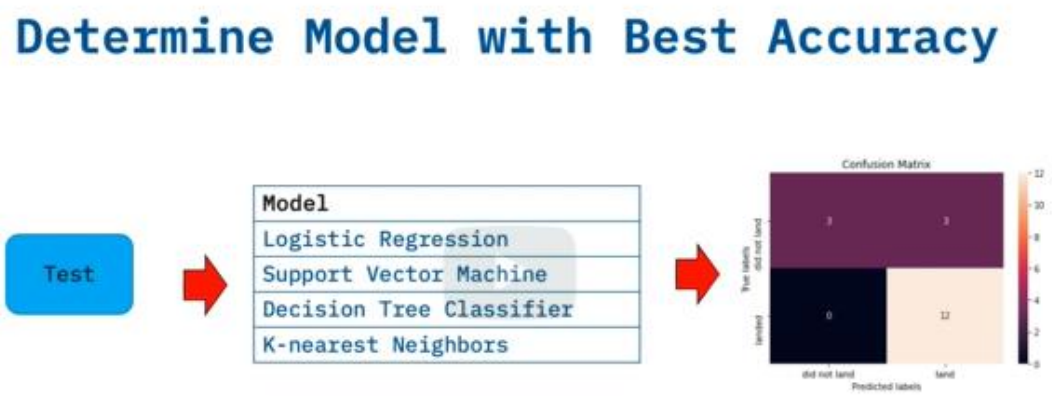
Module Introduction and Learning Objectives

In this section, you will use machine learning to determine if the first stage of Falcon 9 will land successfully. You will split your data into training data and test data to find the best Hyperparameter for SVM, Classification Trees, and Logistic Regression. Then find the method that performs best using test data.

Learning Objectives

Train different classification models.
Split the data into training testing data.
Hyperparameter grid search.
Use your machine learning skills to build a predictive model to help a business function more efficiently.

We will build a machine learning pipeline to predict if the first stage of the Falcon 9 lands successfully.
This will include: Preprocessing, allowing us to standardize our data, and Train_test_split, allowing us to split our data into training and testing data, We will train the model and perform Grid Search, allowing us to find the hyperparameters that allow a given algorithm to perform best.



Using the best hyperparameter values, we will determine the model with the best accuracy using the training data. You will test Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbors. Finally, we will output the confusion matrix.

10.4.Predictive Analysis (Classification)

Seongjoo Brenden Song edited this page on Nov 7, 2021 · 2 revisions

In this module, you will use machine learning to determine if the first stage of Falcon 9 will land successfully. You will split your data into training data and test data to find the best Hyperparameter for SVM, Classification Trees, and Logistic Regression. Then find the method that performs best using test data.

Learning Objectives

- Split the data into training testing data.
- Train different classification models.
- Hyperparameter grid search.
- Use your machine learning skills to build a predictive model to help a business function more efficiently.

Complete the Machine Learning Prediction lab.ipynb

IBM-Data-Science-Professional-Certificate/10_Applied_Data_Science_Capstone/8. Complete the Machine Learning Prediction lab.ipynb

https://github.com/brendensong/IBM-Data-Science-Professional-Certificate/blob/main/10_Applied%20Data%20Science%20Capstone/8.%20Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Space X Falcon 9 First Stage Landing Prediction

Assignment: Machine Learning Prediction

Estimated time needed: **60** minutes

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. In this lab, you will create a machine learning pipeline to predict if the first stage will land given the data from the preceding labs.

Several examples of an unsuccessful landing are shown here:

Most unsuccessful landings are planed. Space X; performs a controlled landing in the oceans.

Objectives

Perform exploratory Data Analysis and determine Training Labels

- create a column for the class
- Standardize the data
- Split into training data and test data

-Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

- Find the method performs best using test data

Import Libraries and Define Auxiliary Functions

We will import the following libraries for the lab

In [2]:
```
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collectio
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use this in our plotter function to plot da
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statisti
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find the best one
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

This function is to plot the confusion matrix.

In [3]:
```
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'landed'])
```

Load the dataframe

Load the data

In [4]:
```
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

# If you were unable to complete the previous lab correctly you can uncomment and load this csv

# data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_p

data.head()
```

Out[4]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False

LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
NaN	1.0	0	B0003	-80.577366	28.561857	0
NaN	1.0	0	B0005	-80.577366	28.561857	0
NaN	1.0	0	B0007	-80.577366	28.561857	0
NaN	1.0	0	B1003	-120.610829	34.632093	0
NaN	1.0	0	B1004	-80.577366	28.561857	0

In [5]:

```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv')

# If you were unable to complete the previous lab correctly you can uncomment and load this csv

# X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part

X.head(100)
```

Out[5]:

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	0.0
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...
85	86.0	15400.000000	2.0	5.0	2.0	0.0	0.0	0.0
86	87.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0
87	88.0	15400.000000	6.0	5.0	5.0	0.0	0.0	0.0
88	89.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0
89	90.0	3681.000000	1.0	5.0	0.0	0.0	0.0	0.0

90 rows × 83 columns

Orbit_HEO	Orbit_ISS	...	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062	GridFins_False	GridFins_True	Reused_False	Reused_True	Legs_False	Legs_True
0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.0	1.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...
0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
0.0	0.0	...	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0
0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0

TASK 1

Create a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y,make sure the output is a Pandas series (only one bracket df['name of column']).

In [6]:
```
Y = data['Class'].to_numpy()
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

In [7]:
```
# students get this
transform = preprocessing.StandardScaler()
```

In [8]:
```
X = transform.fit_transform(X)
```

We split the data into training and testing data using the function train_test_split. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function GridSearchCV.

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [9]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

In [10]:
```
Y_test.shape
```

Out[10]:
```
(18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

In [11]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

In [12]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv = 10)
logreg_cv.fit(X_train, Y_train)
```

Out[12]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

In [13]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 5

Calculate the accuracy on the test data using the method `score` :

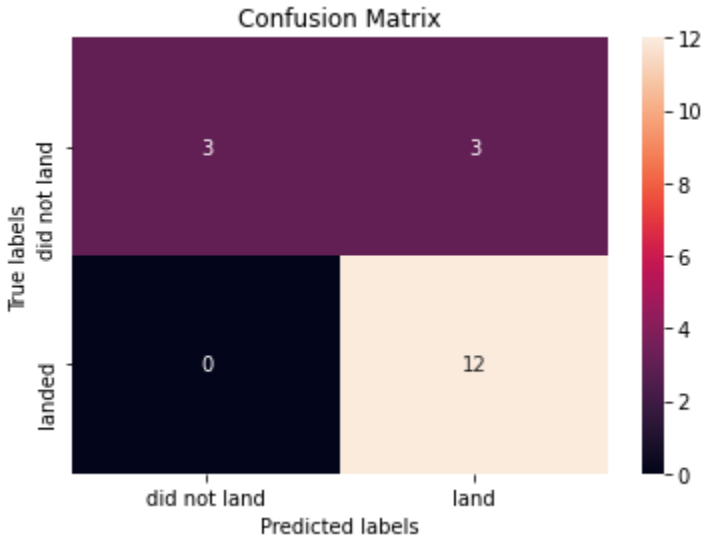
In [14]:
```
# for comparing the accuracy by the methods
methods = []
accuracy = []

methods.append('Logistic regression')
accuracy.append(logreg_cv.score(X_test, Y_test))
```

In [15]:
```
print("test set accuracy :",logreg_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

Lets look at the confusion matrix:

In [16]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

TASK 6

Create a support vector machine object then create a GridSearchCV object svm_cv with cv - 10. Fit the object to find the best parameters from the dictionary parameters.

In [17]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

In [18]:
```
svm_cv = GridSearchCV(svm, parameters, cv = 10)
svm_cv.fit(X_train, Y_train)
```

Out[18]:
```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

In [19]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

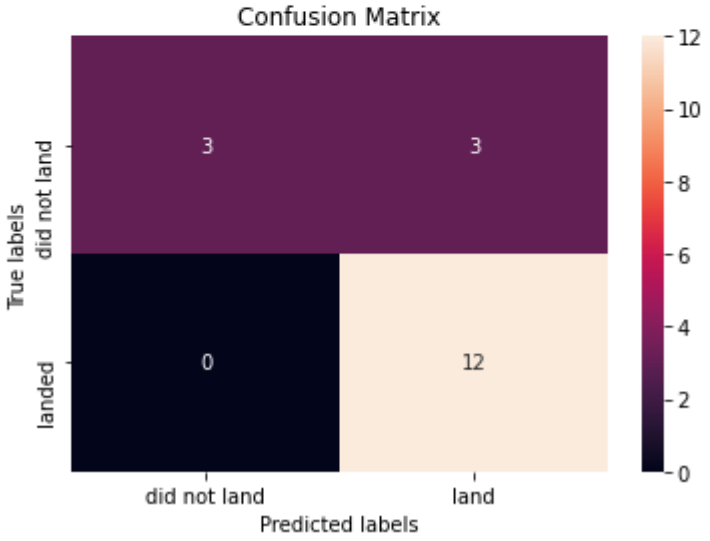
Calculate the accuracy on the test data using the method score:

In [20]:
```
methods.append('Support vector machine')
accuracy.append(svm_cv.score(X_test, Y_test))
print("test set accuracy :",svm_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

We can plot the confusion matrix

In [21]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 8

Create a decision tree classifier object then create a GridSearchCV object tree_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

In [22]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

In [23]:
```
tree_cv = GridSearchCV(tree,parameters,cv=10)
tree_cv.fit(X_train, Y_train)
```

Out[23]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

In [24]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min
_samples_split': 5, 'splitter': 'random'}
accuracy : 0.875
```

TASK 9

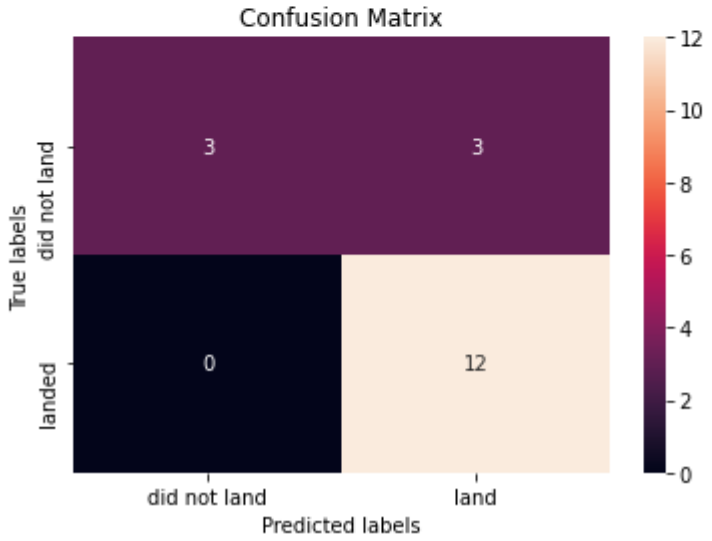
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [25]:
```
methods.append('Decision tree classifier')
accuracy.append(tree_cv.score(X_test, Y_test))
print("test set accuracy :",tree_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

We can plot the confusion matrix

In [26]:
```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 10

Create a k nearest neighbors object then create a GridSearchCV object knn_cv with cv = 10. Fit the object to find the best parameters from the dictionary parameters.

In [27]:
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

In [28]:
```
knn_cv = GridSearchCV(KNN,parameters,cv=10)
knn_cv.fit(X_train, Y_train)
```

Out[28]:
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

In [29]:
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

TASK 11

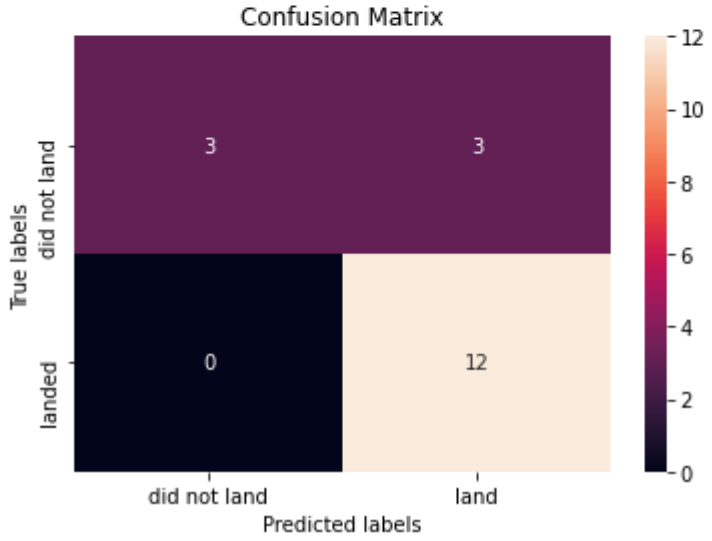
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [30]:
```
methods.append('K nearest neighbors')
accuracy.append(knn_cv.score(X_test, Y_test))
print("test set accuracy :",knn_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

We can plot the confusion matrix

In [31]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

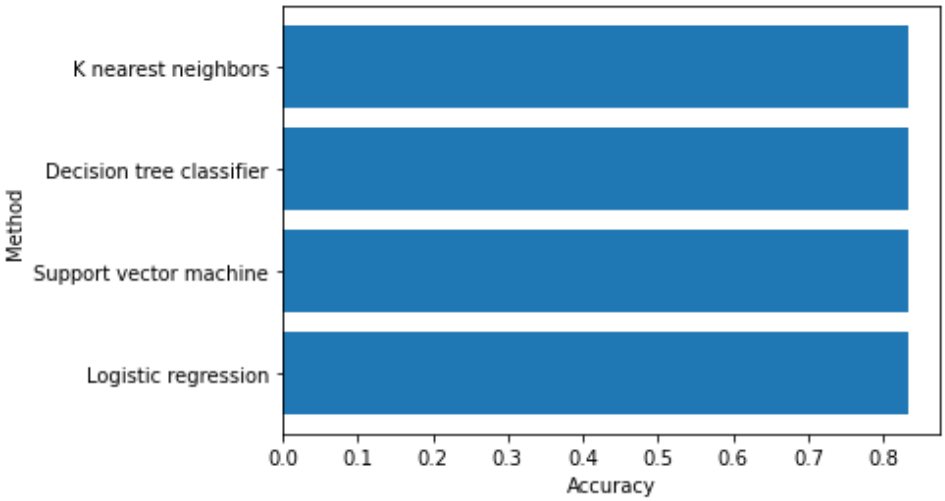


TASK 12

Find the method performs best:

In [33]:

```
import numpy as np
import matplotlib.pyplot as plt

plt.barh(methods, accuracy)
plt.xlabel('Accuracy')
plt.ylabel('Method')
plt.show()
```



After comparing the accuracy of the above methods, all return the same accuracy for the test data.

Graded Quiz: Predictive Analysis

Bookmark this page

Quiz due May 13, 2022 18:10 +08

Question 1

1/1 point (graded)

How many records were there in the test sample ?

18 ✔

18

Submit You have used 1 of 2 attempts

✔ Correct (1/1 point)

Question 2

1/1 point (graded)

For Support Vector Machines, what kernel has the best result on the validation dataset?

- ○ rbf
- ○ linear
- ◉ sigmoid ✔

Submit You have used 1 of 2 attempts

✔ Correct (1/1 point)

Question 3

1/1 point (graded)

After selecting the best hyperparameters for the decision tree classifier using the validation data, what was the accuracy achieved on the test data?

- ○ 93.33%
- ○ 73.33%
- ◉ 83.33%

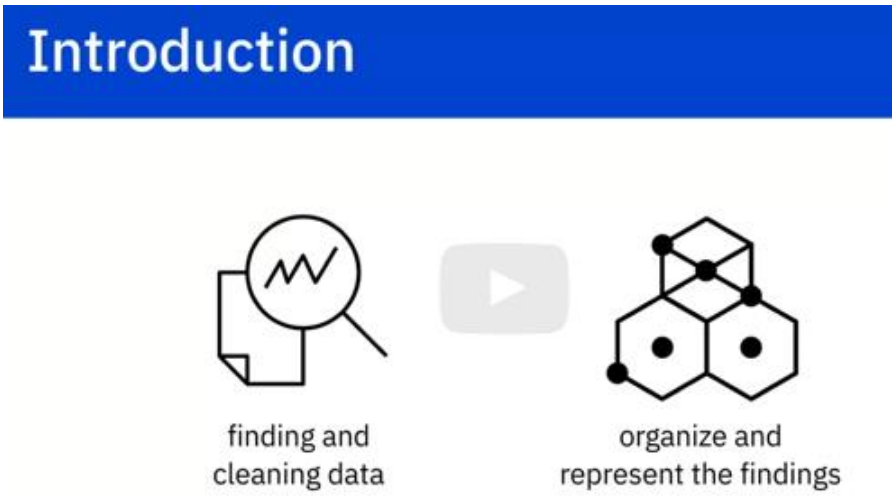
✔

Answer
Correct: Correct.

Submit You have used 1 of 2 attempts Save Show answ

✔ Correct (1/1 point)

Elements of a Successful Data Findings Report



While finding and cleaning data is an important first step in data analysis, a concept can be lost if you are not able to organize and represent the findings effectively to your audience. In this video, you will learn how to represent your findings by focusing on specific elements to create a successful data findings report.

After the data has been collected, cleaned, and organized the work of interpretation begins. You are now able to obtain a complete view of the data and hopefully, answer the questions that were formed before starting the analysis. Now, you typically begin to compose a findings report that explains what was learned. Depending on the stakeholders and how they receive the information, your report could vary in form. This could include a paper style report, a slideshow presentation, or maybe even both.

1.

By completing an outline, you can then get a complete picture and begin to write in a precise but simple manner.

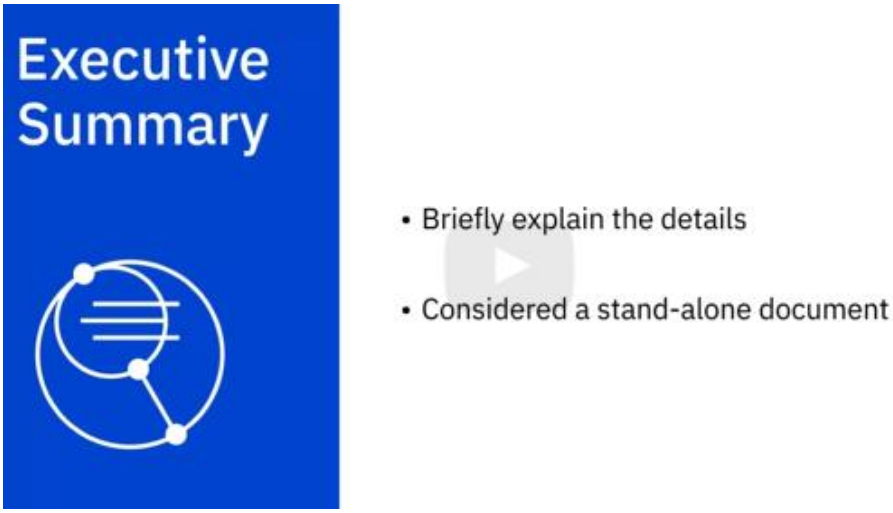
While there are many different formats for creating a data-driven presentation, we have created a simple outline that is easy to follow yet effective. When creating your outline always remember to structure it towards your audience and create a presentation that is appropriate for your situation.

2.

2.1

You first begin with your cover page. This beginning section will have the title of your presentation, your name, and then the date.

2.2



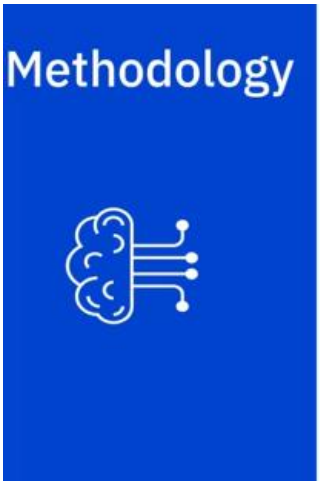
The first step in creating your report is properly creating an executive summary. This summary will briefly explain the details of the project and should be considered a stand-alone document. This information is taken from the main points of your report and while it is acceptable to repeat information, no new information is presented.

2.3

- Nature of the analysis
- States the problem
- States questions for analysis

The next section, after the table of contents, is the introduction. The introduction explains the nature of the analysis, states the problem, and gives the questions that were to be answered by performing the analysis.

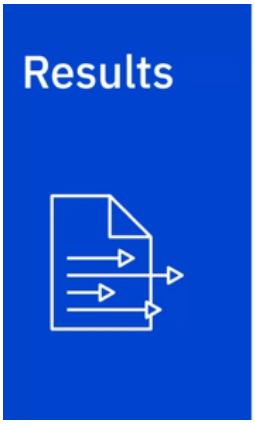
2.4



The next section is methodology. Methodology explains the data sources that were used in the analysis and outlines the plan for the collected data. For example, was the cluster or regression method used to analyze the data?

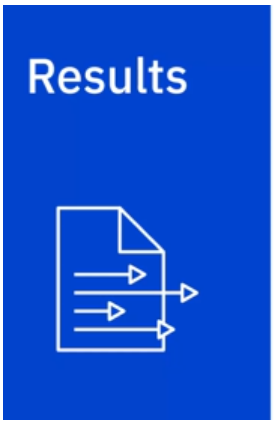
2.5

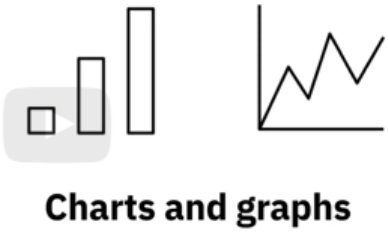
2.5.1



This section goes into the detail of the data collection, how it was organized, and how it was analyzed.

2.5.2





This portion would also contain the charts and graphs that would substantiate the results and call attention to the more complex or crucial findings. By providing this interpretation of data, you are able to give a detailed explanation to the audience and how it relates to the problem that was stated in the introduction.

2.6

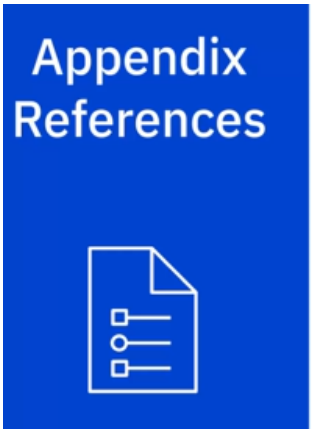
Next is the discussion of the report findings and implications. For this section you would begin to engage the audience with a discussion of your implications that were drawn from the research. For example, let's say you were conducting research for top programming languages for college graduates. Would you find they need to learn multiple languages to remain competitive in the job market, or would one language always reign supreme?

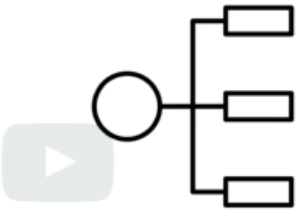
2.7

Conclusion of the report findings

This final section should reiterate the problem given in the introduction and gives an overall summary of the findings. It would also state the outcome of the analysis and if any other steps would be taken in the future.

2.8





Information that didn't fit in the report

This section would contain information that really didn't fit in the main body of the report, but you deemed it was still important enough to include. This type of information could include locations where the raw data was collected or other details such as resources, acknowledgements or references.

Conclusion

About the important elements in creating a successful data findings report.

10.5. Present Your Data Driven Insights

Seongjoo Brenden Song edited this page on Nov 7, 2021 · 3 revisions

In this module, you will compile all of your activities into one place and deliver your data-driven insights to determine if the first stage of Falcon 9 will land successfully.

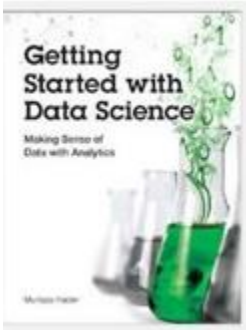
Learning Objectives

- Learn how to structure and build your data-findings report.
- Submit your final report for peer review.
- Review the work submitted by your peers.

- Reading: Structure Of A Report
- Capstone Project Presentation

Course Text Book: 'Getting Started with Data Science' Publisher: IBM Press; 1 edition (Dec 13 2015) Print.

Author: Murtaza Haider



Prescribed Reading: Chapter 3 Pg. 60-62

The Report Structure

Before starting the analysis, think about the structure of the report. Will it be a brief report of five or fewer pages, or will it be a longer document running more than 100 pages in length? The structure of the report depends on the length of the document. A brief report is more to the point and presents a summary of key findings. A detailed report incrementally builds the argument and contains details about other relevant works, research methodology, data sources, and intermediate findings along with the main results.

I have reviewed reports by leading consultants including Deloitte and McKinsey. I found that the length of the reports varied depending largely on the purpose of the report. Brief reports were drafted as commentaries on current trends and developments that attracted public or media attention. Detailed and comprehensive reports offered a critical review of the subject matter with extensive data analysis and commentary. Often, detailed reports collected new data or interviewed industry experts to answer the research questions.

Even if you expect the report to be brief, sporting five or fewer pages, I recommend that the deliverable follow a prescribed format including the cover page, table of contents, executive summary, detailed contents, acknowledgments, references, and appendices (if needed).

I often find the cover page to be missing in documents. It is not the inexperience of undergraduate students that is reflected in submissions that usually miss the cover page. In fact, doctoral candidates also require an explicit reminder to include an informative cover page. I hasten to mention that the business world sleuths are hardly any better. Just search the Internet for reports and you will find plenty of reports from reputed firms that are missing the cover page.

At a minimum, the cover page should include the title of the report, names of authors, their affiliations, and contacts, the name of the institutional publisher (if any), and the date of publication. I have seen numerous reports missing the date of publication, making it impossible to cite them without the year and month of publication. Also, from a business point of view, authors should make it easier for the reader to reach out to them. Having contact details at the front makes the task easier.

"A table of contents (ToC)" is like a map needed for a trip never taken before. You need to have a sense of the journey before embarking on it. A map provides a visual proxy for the actual travel with details about the landmarks that you will pass by in your trip. The ToC with main headings and lists of tables and figures offers a glimpse of what lies ahead in the document. Never shy away from including a ToC, especially if your document, excluding cover page, table of contents, and references, is five or more pages in length.

Even for a short document, I recommend an "abstract" or an "executive summary". Nothing is more powerful than explaining the crux of your arguments in three paragraphs or less. Of course, for larger documents running a few hundred pages, the executive summary could be longer. An "introductory section" is always helpful in setting up the problem for the reader who might be new to the topic and who might need to be gently introduced to the subject matter before being immersed in intricate details. A good follow-up to the introductory section is a review of available relevant research on the subject matter. The length of the literature review section depends upon how contested the subject matter is. In instances where the vast majority of researchers have concluded in one direction, the literature review could be brief with citations for only the most influential authors on the subject. On the other hand, if the arguments are more nuanced with caveats aplenty, then you must cite the relevant research to offer adequate context before you embark on your analysis. You might use the literature review to highlight gaps in the existing knowledge, which your analysis will try to fill. This is where you formally introduce your research questions and hypothesis.

In the "methodology" section, you introduce the research methods and data sources you used for the analysis. If you have collected new data, explain the data collection exercise in some detail. You will refer to the literature review to bolster your choice for variables, data, and methods and how they will help you answer your research questions.

The results section is where you present your empirical findings. Starting with descriptive statistics (**see Chapter 4, "Serving Tables"**) and illustrative graphics (**see Chapter S, "Graphic Details" for plots and Chapter 10, "Spatial Data Analytics" for maps**), you will move toward formally testing your hypothesis (**see Chapter 6, "Hypothetically Speaking"**).

In case you need to run statistical models, you might turn to regression models (**see Chapter 7, "Why Tall Parents Don't Have Even Taller Children"**) or categorical analysis (**see Chapters 8, "To Be or Not to Be" and 2., "Categorically Speaking About Categorical Data"**). If you are working with time-series data, you can turn to Chapter 11, **Doing Serious Time with Time Series.** You can also report results from other empirical techniques that fall under the general rubric of data mining (**see Chapter 12, "Data Mining for Gold"**). Note that many

reports in the business sector present results in a more palatable fashion by holding back the statistical details and relying on illustrative graphics to summarize the results.

The results section is followed by the discussion section, where you craft your main arguments by building on the results you have presented earlier.

The "discussion section" is where you rely on the power of narrative to enable numbers to communicate your thesis to your readers. You refer the reader to the research question and the knowledge gaps you identified earlier. You highlight how your findings provide the ultimate missing piece to the puzzle.

Of course, not all analytics return a smoking gun. At times, more frequently than I would like to acknowledge, the results provide only a partial answer to the question and that, too, with a long list of caveats.

In the "conclusion" section, you generalize your specific findings and take on a rather marketing approach to promote your findings so that the reader does not remain stuck in the caveats that you have voluntarily outlined earlier. You might also identify future possible developments in research and applications that could result from your research. What remains is housekeeping, including a list of references, the acknowledgment section (**acknowledging the support of those who have enabled your work is always good**), and "appendices", if needed.

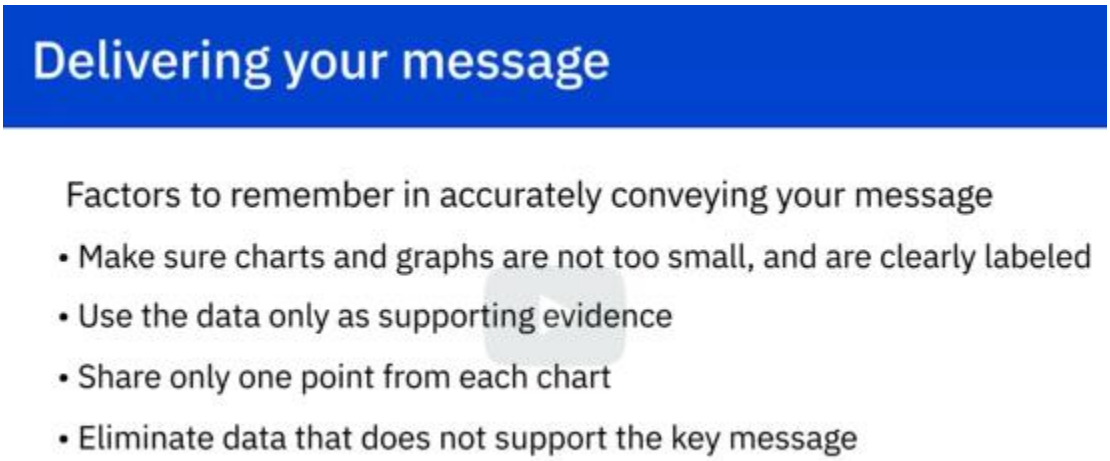
Have You Done Your Job as a Writer?

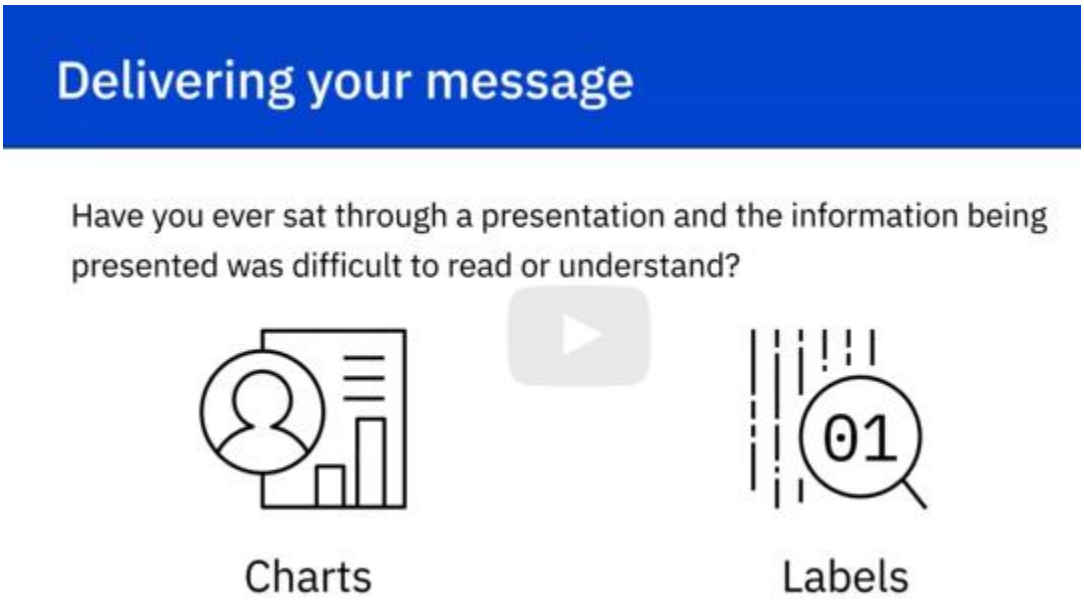
As a data scientist, you are expected to do thorough analysis with the appropriate data, deploying the appropriate tools. As a writer, you are responsible for communicating your findings to the readers. Transport Policy, a leading research publication in transportation planning, offers a checklist for authors interested in publishing with the journal. The checklist is a series of questions authors are expected to consider before submitting their manuscripts to the journal. I believe the checklist is useful for budding data scientists and, therefore, I have reproduced it verbatim for their benefit.

- Have you told readers, at the outset, what they might gain by reading your paper?
- Have you made the aim of your work clear?
- Have you explained the significance of your contribution?
- Have you set your work in the appropriate context by giving sufficient background (including a complete set of relevant references) to your work?
- Have you addressed the question of practicality and usefulness?
- Have you identified future developments that might result from your work?
- Have you structured your paper in a clear and logical fashion?

Best Practices For Presenting Your Findings

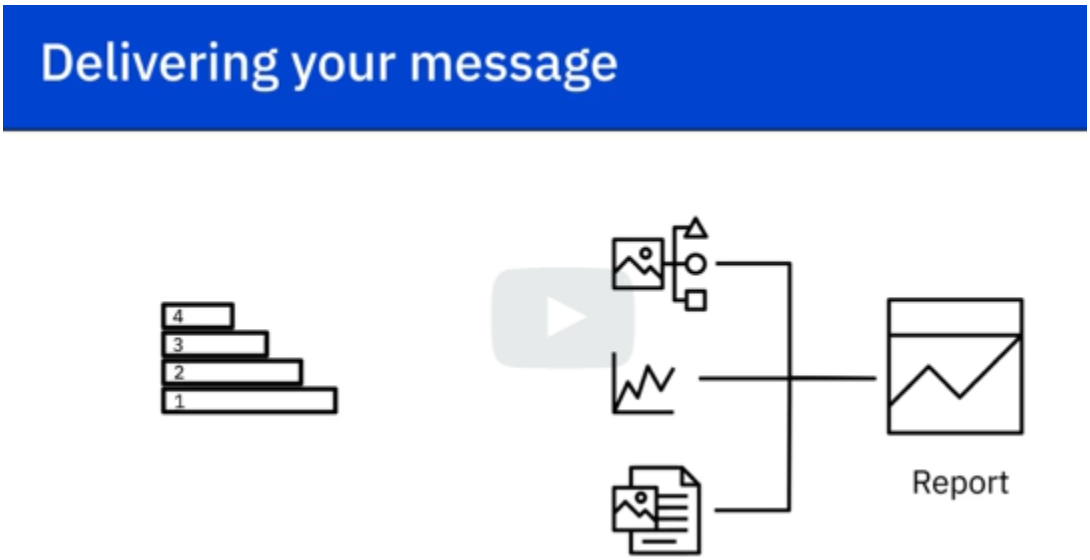
Okay, you've spent weeks, maybe months, studying the data and the time has come to report your findings. The questions have been answered, and you feel good about the story. How will you speak to your audience so they leave with the intended message? In this video, learn how to present your findings in a way that will engage and keep the attention of your audience.





Make sure charts and graphs are not too small and are clearly labeled, Use the data only as supporting evidence, Share only one point from each chart or graph, and Eliminate data that does not support the key message. Have you ever sat through a presentation and the information being presented was difficult to read

or understand? While this may seem apparent, small charts and labels can be easily overlooked. Make sure to test the visualizations by sitting at different distances, similar to your audience, and if the data cannot be seen clearly then maybe a redesign should be considered.



When preparing the report, you may feel the only way to explain the findings is to pack the slides with data. While this may seem sensible as a data analyst, your audience will probably not appreciate the intricacies of the data and just see a pile of numbers. To resolve this issue, begin by forming the key messages that need to be conveyed to the audience and build the story around these messages. After forming the outline, go back and insert the data to support your findings. By not relying heavily on the data and using this method to create the presentation, you will create a story that is engaging and interesting to your audience.

Presenting your data using charts and graphs is the best way to get your message across to the audience, however, if you are supplying too much information it can be confusing. For example, looking at this pie chart, can you decipher what the key message is, and what the presenter is trying to convey? In the example, the

chart has so much information it is hard to determine what point the presenter is trying to make and what the focus should be for the audience. By sticking with one idea and not summarizing multiple points into one visualization, you are able to accurately convey the idea to the audience and avoid any confusion.

Delivering your message

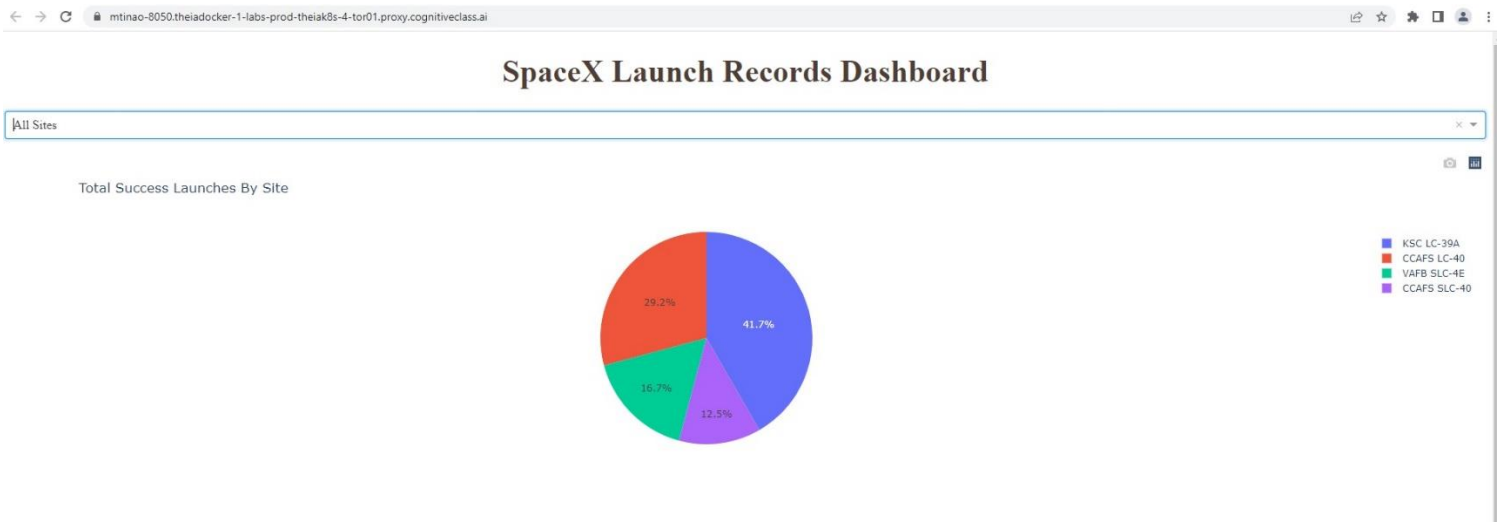
Data analysts can spend months researching data, however, some items that seem interesting to the analyst may not be relevant to the project. Trying to explain every little detail to your audience and not recognizing irrelevant data could damage the key message.

Delivering your message

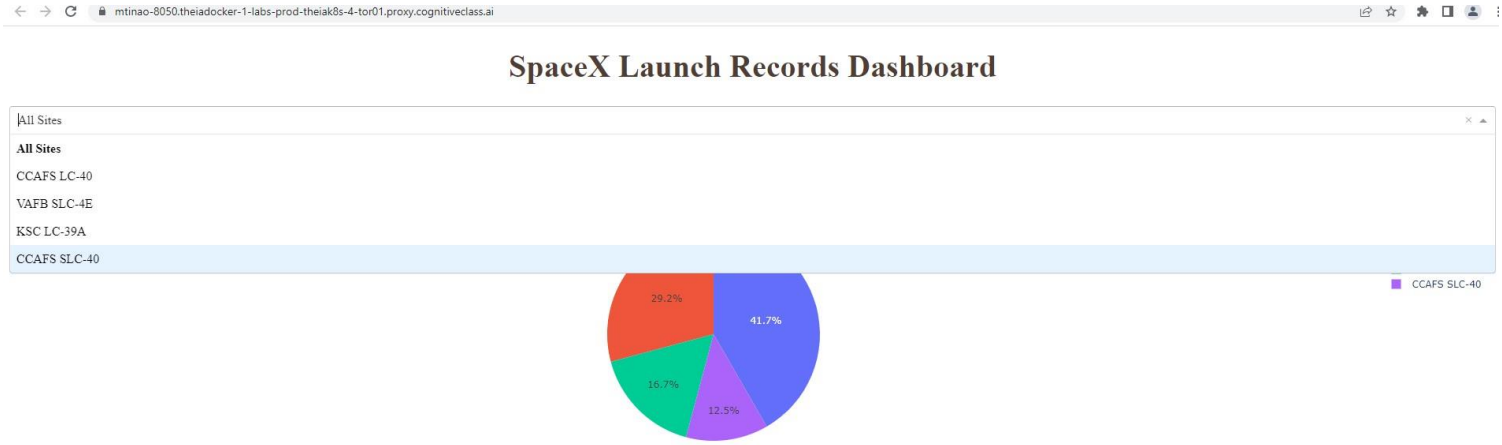
By eliminating this unnecessary data and highlighting only data points that support your key ideas, you will keep the presentation clear and concise.

In this video, we learned about creating a data-driven presentation that will keep the audience engaged and how to deliver a clear and concise message.

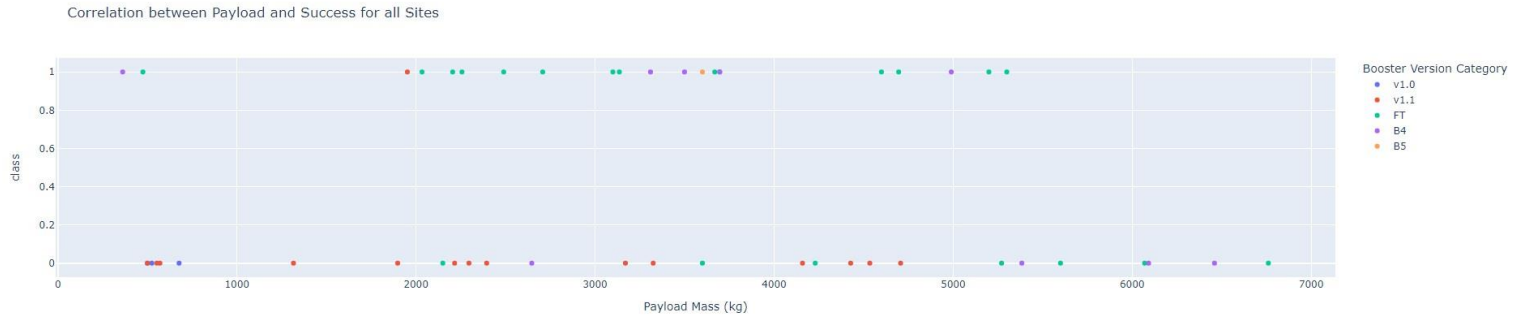
Module 3 Interactive Plotty Dash

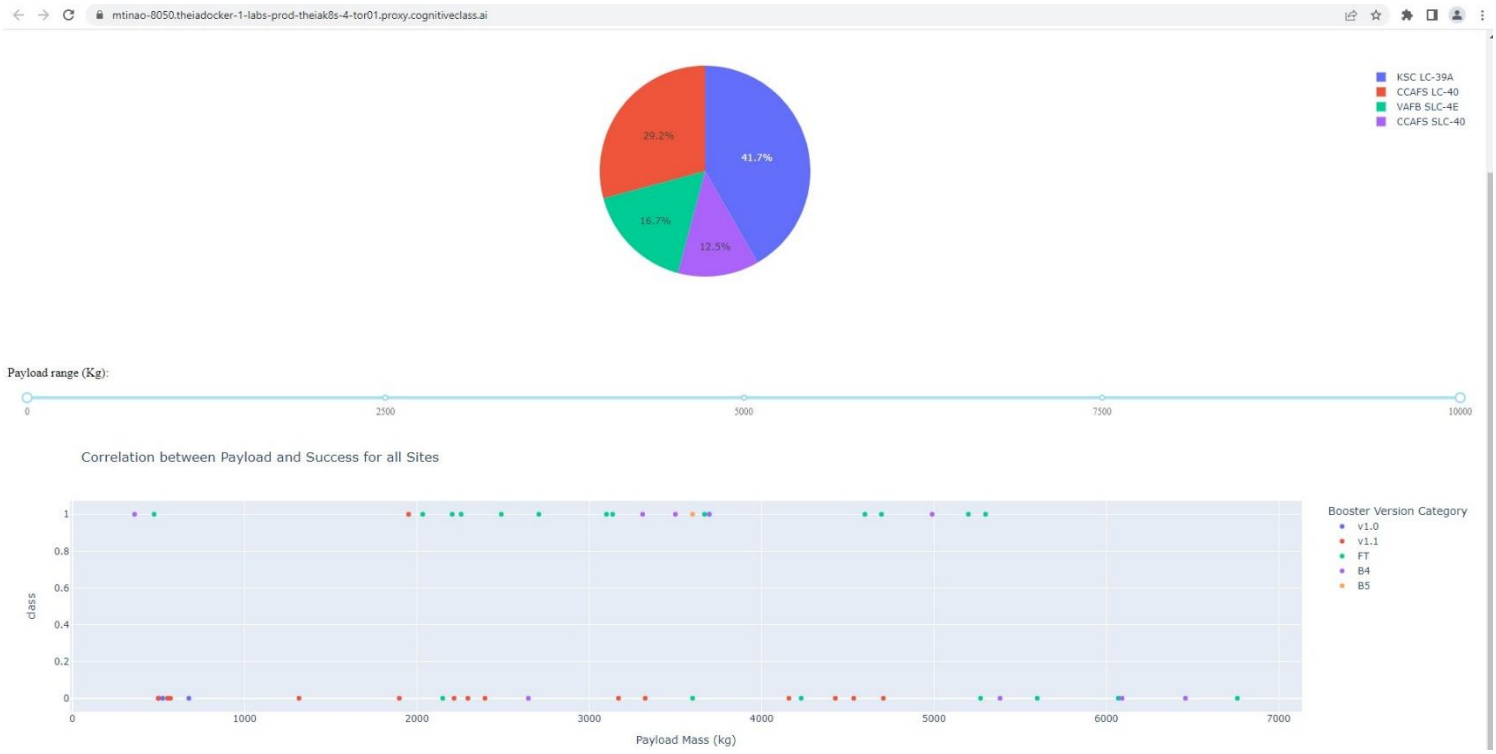


86



Payload range (Kg):





Check Points: Interactive Visual Analytics and Dashboard

Bookmark this page

Question 1

1/1 point (ungraded)

Have you marked all launch sites on a Folium map?

- (•) Yes
- ( ) No

✔

Submit

Question 2

1/1 point (ungraded)

Have you marked the success/failed launches for each site?

- (•) Yes
- ( ) No

✔

Submit

Question 3

1/1 point (ungraded)

Have you calculated the distances between a launch site to its proximities?

- (●) Yes
- ( ) No

✔

Submit

Question 4

1/1 point (ungraded)

Have you completed and added a launch site drop-down input component?

- (●) Yes
- ( ) No

✔

Submit

Question 5

1/1 point (ungraded)

Have you completed and added a callback function to render the required success outcome pie chart?

- (•) Yes
- ( ) No

✔

Submit

Sho

Question 6

1/1 point (ungraded)

Have you completed and added a range slider to choose payload?

- (•) Yes
- ( ) No

✔

Question 7

1/1 point (ungraded)

Have you completed and added callback function to render the payload-outcome scatter plot?

- (•) Yes
- ( ) No

✔

Submit

Graded Quiz: Interactive Visual Analytics and Dashboard

Bookmark this page

Quiz due May 9, 2022 02:10 +08 Completed

Question 1

2.0/2.0 points (graded)

How can you add marking objects such as circles, markers, or lines on a Folium map? (Click all choices that apply) (2 pts)

- [x] map.add_child(object)
- [x] object.add_to(map)
- [ ] add_node(map, object)
- [ ] map.add_to(object)

Submit You have used 1 of 2 attempts Save Show ans

Question 2

2.0/2.0 points (graded)

If you want to add multiple markers with similar coordinates on the Folium map, which Folium plugin you should use? (2 pts)

- ( ) Markers should be add to map directly without any extra layer
- ( ) MarkerGroup
- ( ) MarkerContainer
- (•) MarkerCluster

Submit You have used 1 of 2 attempts Save Show answer

Question 3

2.0/2.0 points (graded)

Which attribute is used to provide available selections (such as a list of launch sites) for a Plotly DropDown input? (2 pts)

- ○ input
- ○ placeholder
- ◉ options
- ○ values

✔

Submit You have used 1 of 2 attempts Save Show ansv

Question 4

2.0/2.0 points (graded)

How can we associate the result of a callback function (like a Ploty figure) to an element defined in the application layout (2 pts)?

- ○ Dash automatically render the result of a callback function
- ○ Using component name
- ◉ Using a unique component id

✔

Submit You have used 1 of 2 attempts Save Show a

Question 5

2.0/2.0 points (graded)

Can we add multiple input components to a dash callback function (2 pts)?

- (●) Yes
- ( ) No

✔

Submit You have used 1 of 1 attempt

Submission Overview and Instructions

The final task of this capstone project is to create a presentation based on the outcomes of all tasks in previous modules and labs. Your presentation will develop into a story of all your data science journey in this project, and it should be compelling and easy to understand.

In the next exercise, you can find a provided PowerPoint template to help you get started to create a report in slides format. However, you are free to add additional slides, charts, and tables.

Note that this presentation will be prepared for your peer-data-scientists whom are eager to understand every technical detail of this project. As such, this presentation will be much more detailed and technical than regular high-level and abstracted presentation for your executive team.

Once you have completed a detailed report, it should be straightforward for you to abstract it into a high-level deck for your executive team and/or stakeholders.

There are a total of 40 points possible for the final assessment, and you will be graded by your peers, who are also completing this assignment.

The main grading criteria will be:

- Uploaded the URL of your GitHub repository including all the completed notebooks and Python files (1 pt)
- Uploaded your completed presentation in PDF format (1 pt)
- Completed the required Executive Summary slide (1 pt)
- Completed the required Introduction slide (1 pt)
- Completed the required data collection and data wrangling methodology related slides (1 pt)
- Completed the required EDA and interactive visual analytics methodology related slides (3 pts)
- Completed the required predictive analysis methodology related slides (1 pt)
- Completed the required EDA with visualization results slides (6 pts)
- Completed the required EDA with SQL results slides (10 pts)
- Completed the required interactive map with Folium results slides (3 pts)
- Completed the required Plotly Dash dashboard results slides (3 pts)
- Completed the required predictive analysis (classification) results slides (6 pts)
- Completed the required Conclusion slide (1 pts)
- Applied your creativity to improve the presentation beyond the template (1 pts)
- Displayed any innovative insights (1 pts)

You will not be judged on:

- Your English language, including spelling or grammatical mistakes
- The content of any text or image(s) or where a link is hyperlinked to

Peer Review: Submit your Work and Review your Peers

The final task of this capstone project is to create a presentation based on the outcomes of all tasks in previous modules and labs. Your presentation will develop into a story of all your data science journey in this project, and it should be compelling and easy to understand.

In the next exercise, you can find a provided PowerPoint template to help you get started to create a report in slides format. However, you are free to add additional slides, charts, and tables.

Note that this presentation will be prepared for your peer-data-scientists whom are eager to understand every technical detail of this project. As such, this presentation will be much more detailed and technical than regular high-level and abstracted presentation for your executive team.

Once you have completed a detailed report, it should be straightforward for you to abstract it into a high-level deck for your executive team and/or stakeholders.

There are a total of 40 points possible for the final assessment, and you will be graded by your peers, who are also completing this assignment.

The main grading criteria will be:

- Uploaded the URL of your GitHub repository including all the completed notebooks and Python files (1 pt)
- Uploaded your completed presentation in PDF format (1 pt)
- Completed the required Executive Summary slide (1 pt)
- Completed the required Introduction slide (1 pt)
- Completed the required data collection and data wrangling methodology related slides (1 pt)
- Completed the required EDA and interactive visual analytics methodology related slides (3 pts)
- Completed the required predictive analysis methodology related slides (1 pt)
- Completed the required EDA with visualization results slides (6 pts)
- Completed the required EDA with SQL results slides (10 pts)
- Completed the required interactive map with Folium results slides (3 pts)
- Completed the required Plotly Dash dashboard results slides (3 pts)
- Completed the required predictive analysis (classification) results slides (6 pts)
- Completed the required Conclusion slide (1 pts)
- Applied your creativity to improve the presentation beyond the template (1 pts)
- Displayed any innovative insights (1 pts)

You will not be judged on:

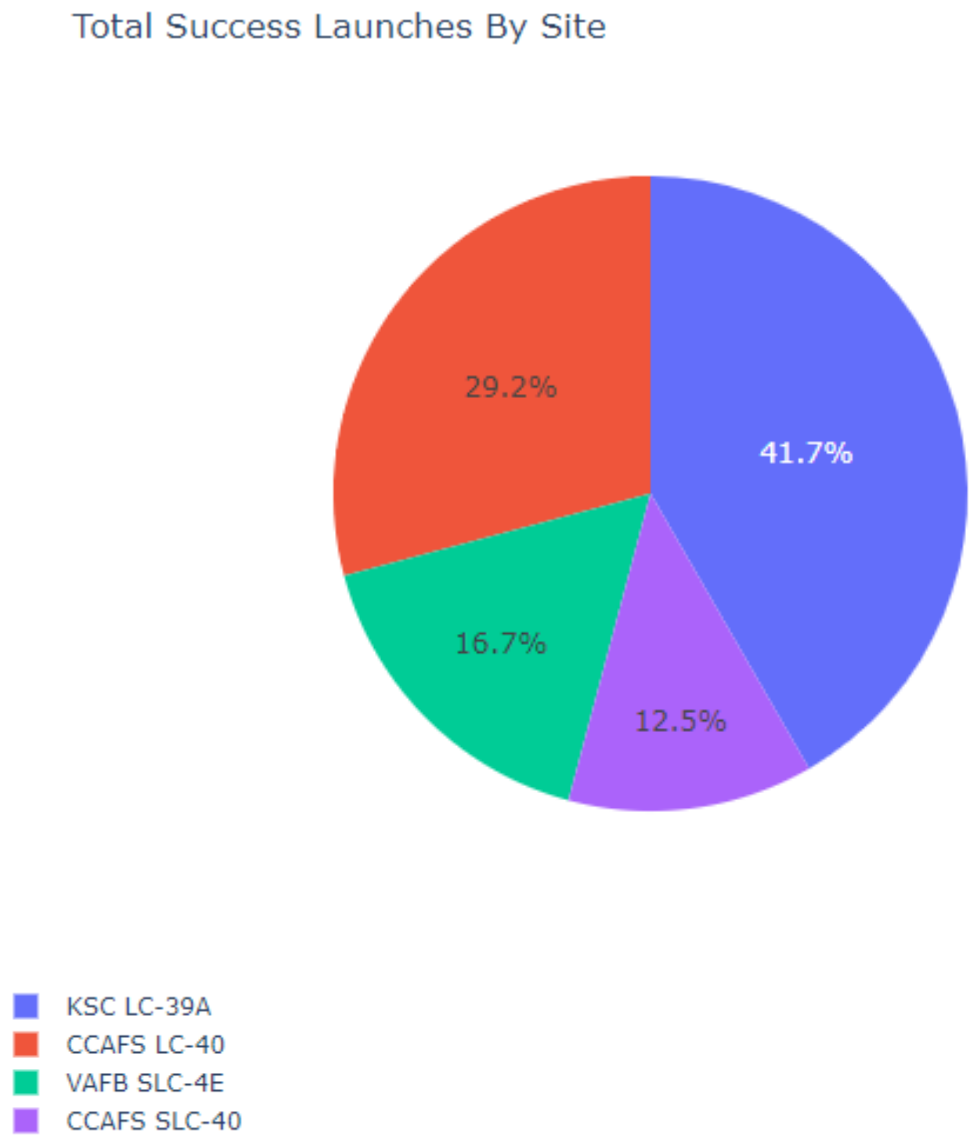
- Your English language, including spelling or grammatical mistakes
- The content of any text or image(s) or where a link is hyperlinked to

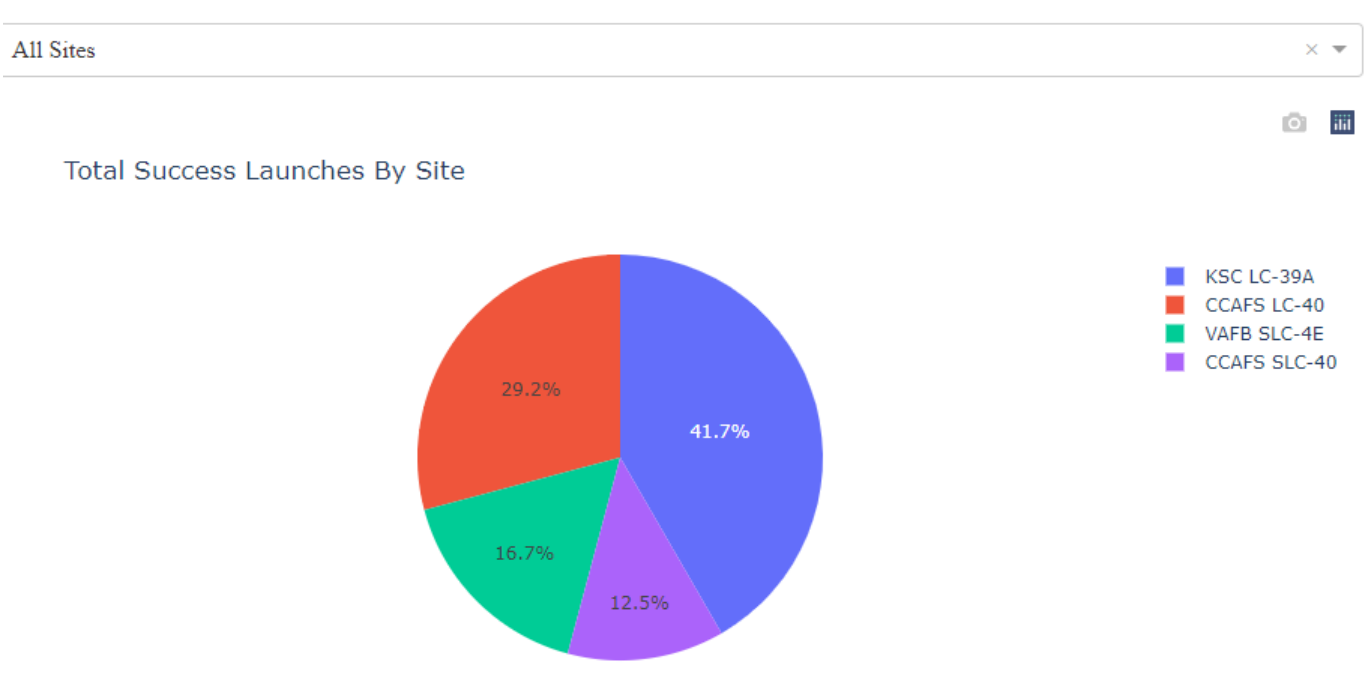
Capstone Project Presentation

Go to IBM DS10 Data Science and Machine Learning capstone Project folder and look for Capstone data Science Project (pdf copy)

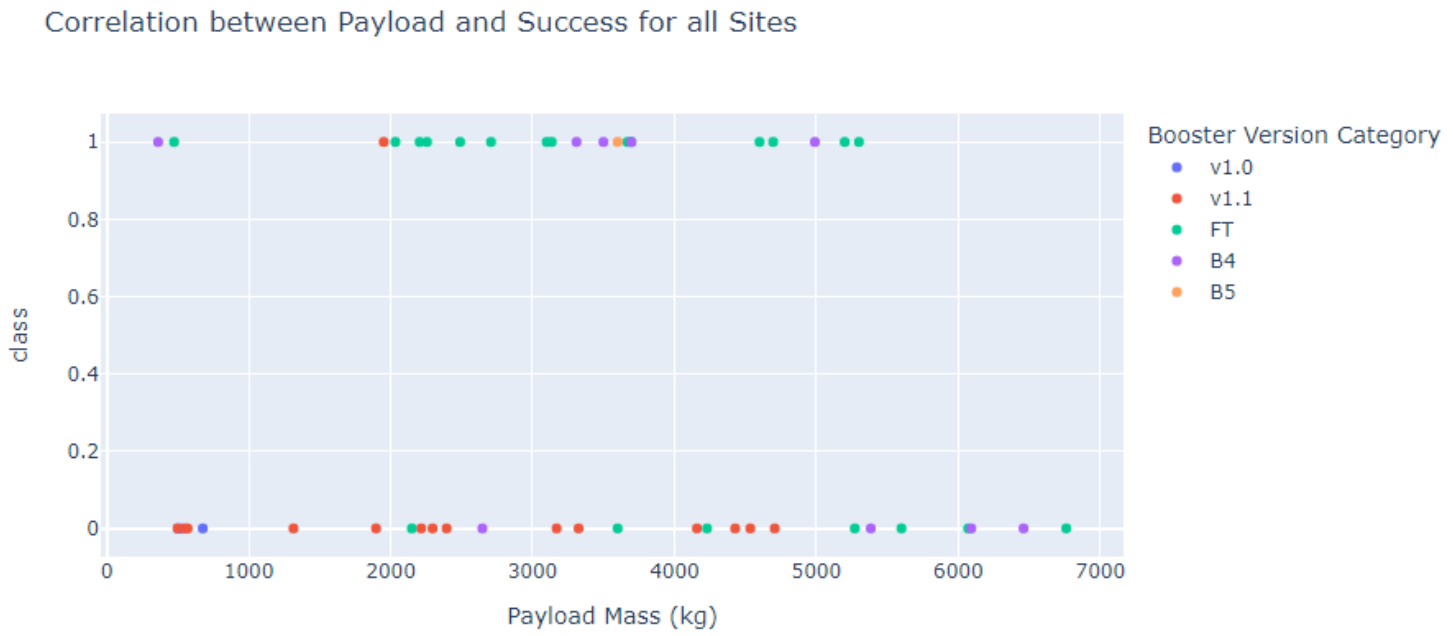
SpaceX Launch Records Dashboard

All Sites



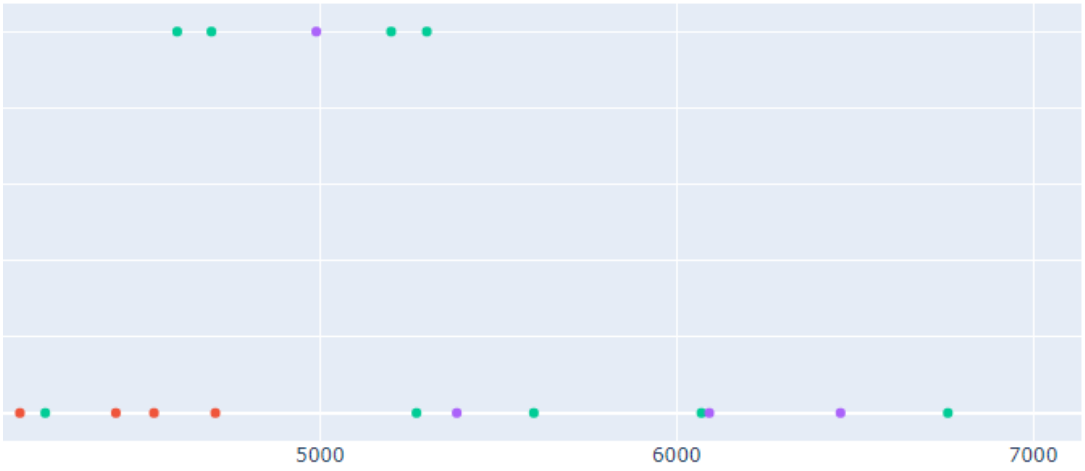


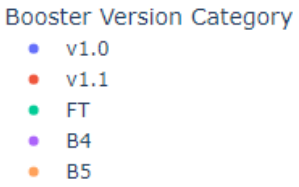
Payload range (Kg):



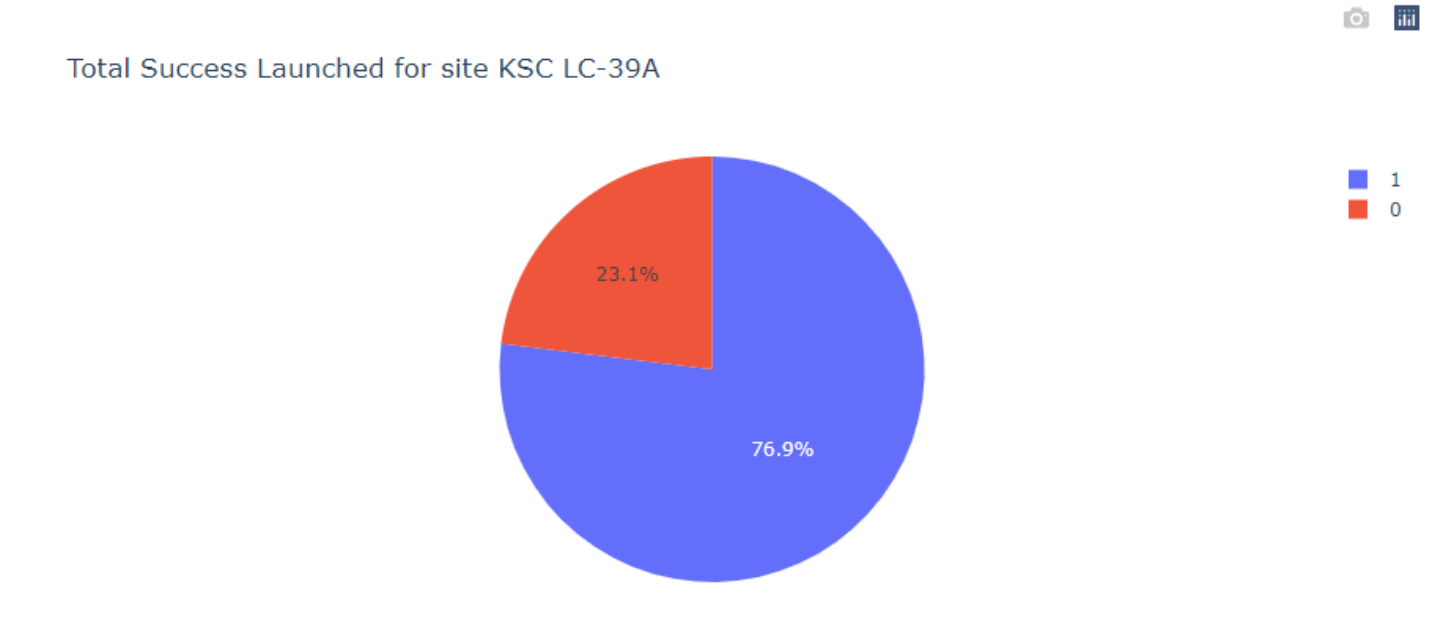
Payload range (Kg):

Start

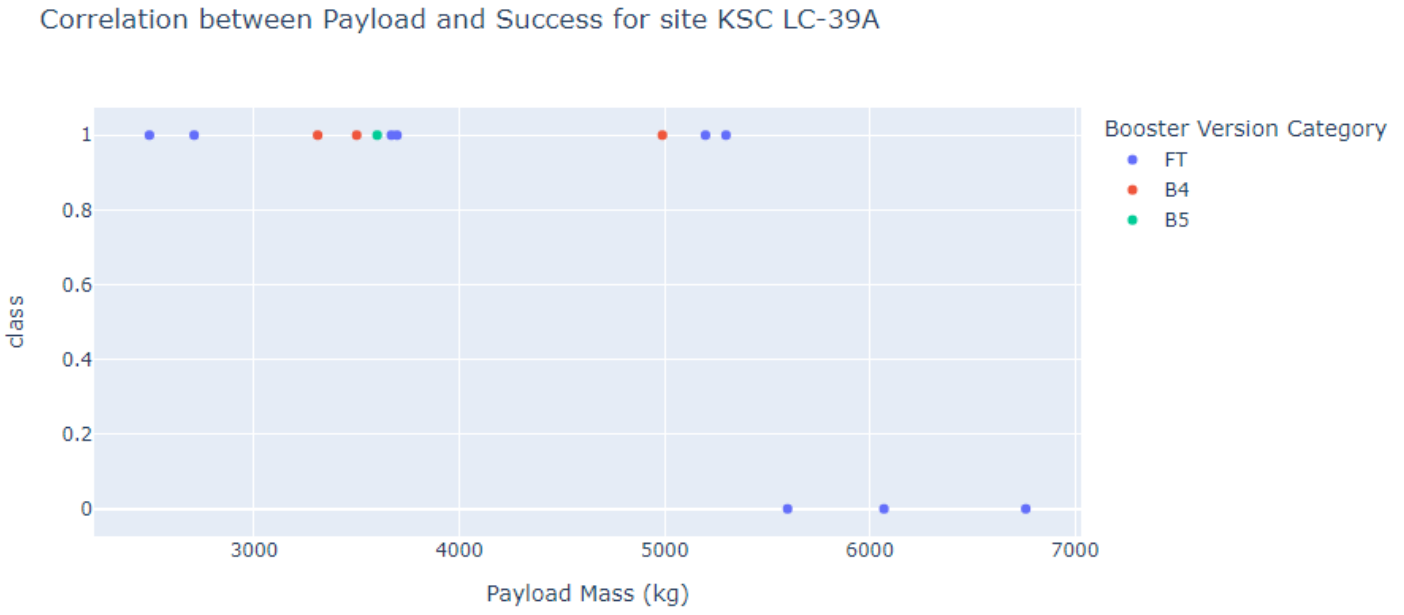


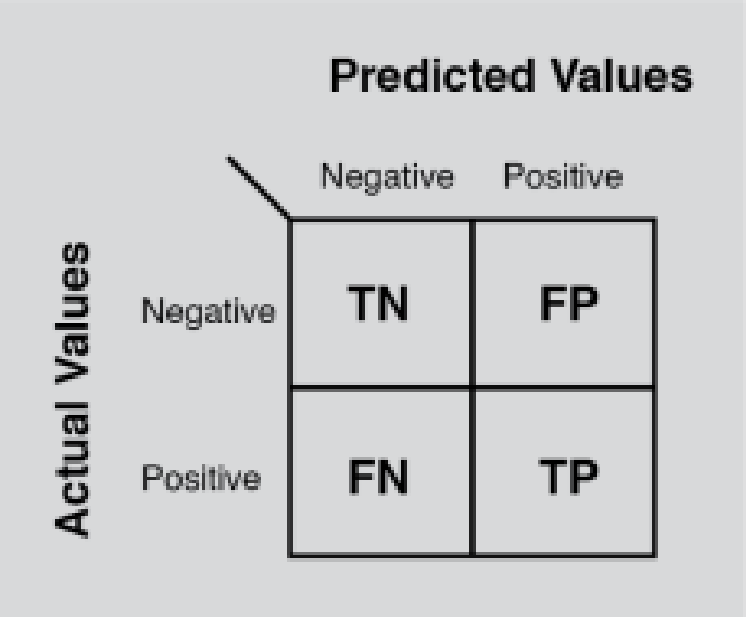


KSC LC-39A



Payload range (Kg):





Your Upload

Files that were uploaded by you:

PDF DS&ML Capstone Project (MMSTinao DS&ML Capstone Project.pdf)

Assessments of Your Response

Status

You have successfully completed this problem and received a 39/40. The grade for this problem is determined by the median score of your Peer Assessments.

Response to prompt 1 1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

Response to prompt 2 1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 2

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 3

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 4

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 5

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 6

3 / 3 POINTS

PEER MEDIAN GRADE - 3 POINTS

Correct

PEER 1 - CORRECT

▼ Response to prompt 7

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▼ Response to prompt 8

6 / 6 POINTS

PEER MEDIAN GRADE - 6 POINTS

Correct

PEER 1 - CORRECT

▼ Response to prompt 9

10 / 10 POINTS

PEER MEDIAN GRADE - 10 POINTS

Correct

PEER 1 - CORRECT

▼ Response to prompt 10

2 / 3 POINTS

PEER MEDIAN GRADE - 2 POINTS

Partially correct

PEER 1 - PARTIALLY CORRECT

▼ Response to prompt 11

3 / 3 POINTS

PEER MEDIAN GRADE - 3 POINTS

Correct

PEER 1 - CORRECT

▼ Response to prompt 12

6 / 6 POINTS

PEER MEDIAN GRADE - 6 POINTS

Correct

PEER 1 - CORRECT

▼ Response to prompt 13

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▾ Response to prompt 14

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▾ Response to prompt 15

1 / 1 POINTS

PEER MEDIAN GRADE - 1 POINT

Correct

PEER 1 - CORRECT

▾ Additional comments on your response

PEER 1

The presentation looks great, congratulations. The only thing missing were all the requiered folium maps screenshots

▾ Provide feedback on peer assessments

Course staff will be able to see any feedback that you provide here when they review course records.

Select the statements below that best reflect your experience with peer assessments.

- [x] **These assessments were useful.**
- [ ] **These assessments were not useful.**
- [ ] **I disagree with one or more of the peer assessments of my response.**
- [ ] **Some comments I received were inappropriate.**

Provide feedback on the grade or comments that you received from your peers.

Thank you very much! Sorry, I missed and forgot about the 2 slides of maps screenshots.