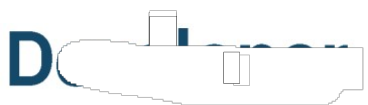




SpaceX-Falcon9

Reshma R

21-12-2022



SKILLS NETWORK

OUTLINE



Executive Summary

Introduction

Methodology

Results

Discussion

Conclusion

Appendix

Developer

SKILLS NETWORK

EXECUTIVE SUMMARY

Data science is the domain of study that deals with vast volumes of data using modern tools and techniques to find unseen patterns, derive meaningful information, and make business decisions. Data science uses machine learning algorithms to build predictive models.

The data used for analysis can come from many different sources and presented in various formats.

INTRODUCTION

In this project 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 dollar; other providers cost upward of 165 million dollar 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.

METHODOLOGY

Data Collection : The process of gathering and analyzing accurate data from various sources to find answers to research problems, trends and probabilities to evaluate possible outcomes.

```
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']:
        if x:
            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.

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

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

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

Developer

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

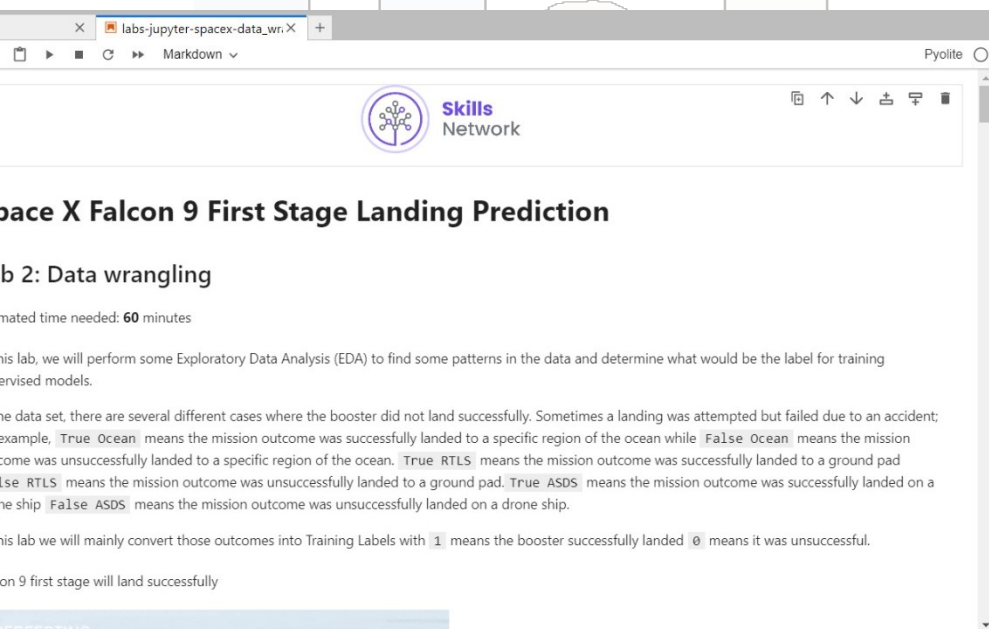
From cores we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, whether the core is reused, whether legs were used, the landing pad used, the block of the core which is a number used to separate version of cores, the number of times this specific core has been reused, and the serial of the core.

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)
```

SKILLS NETWORK

METHODOLOGY

Data Wrangling: The process of removing errors and combining complex data to make them more accessible and easier to analyze.



labs-jupyter-spacex-data_wr.X

Skills Network

Space X Falcon 9 First Stage Landing Prediction

Lab 2: Data wrangling

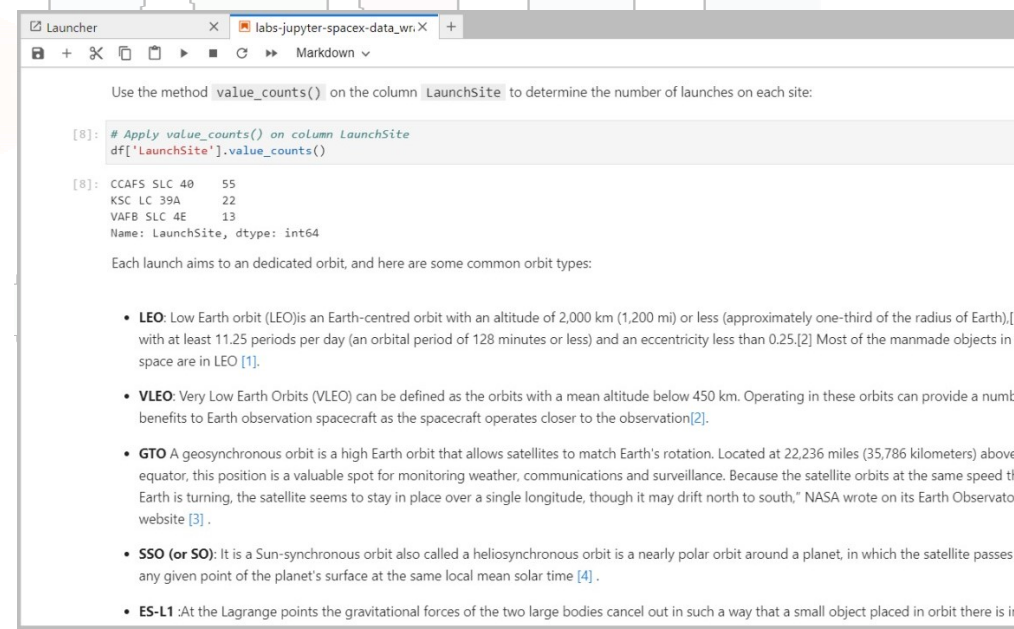
Estimated time needed: 60 minutes

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.

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 while `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 while `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.

On first stage will land successfully



Launcher

labs-jupyter-spacex-data_wr.X

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

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

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

Each launch aims to a 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), 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 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 the equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that 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 Observation 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

Developer

SKILLS NETWORK

Methodology

Data Wrangling:

```
[18]: df.head(5)
```

```
[18]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Success
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	True
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	True
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	True
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	True
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	True

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

Methodology

EDA and interactive visual analytics:

Data scientists implement Exploratory Data Analysis(EDA) tools and techniques to investigate, analyze and summarize the main characteristics of the dataset often utilizing data visualization methodologies.

Interactive data visualization is the use of tools and processes to produce a visual representation of data which can be explored and analyzed directly within the visualization itself.

Methodology

Predictive Analysis: Predictive analytics is a branch of advanced analytics that makes predictions about future outcomes using historical data combined with statistical modeling, data mining techniques and machine learning.

Predictive analytics models are designed to assess historical data, discover patterns, observe trends, and use that information to predict future trends.

EDA with Visualization Results

atory Data Analysis

the SpaceX dataset into a Pandas dataframe and print its summary

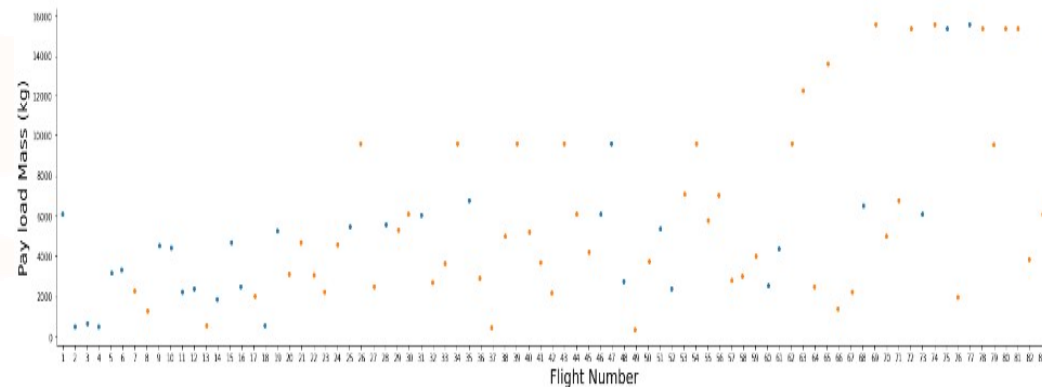
```
csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")
```

Unable to complete the previous lab correctly you can uncomment and load this csv

```
_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0701EN-SkillsNetwork/api/dataset_part_2.csv')
```

er	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	La
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	28.1
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	28.1
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	28.1
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	34.6
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	28.1

```
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 77%.

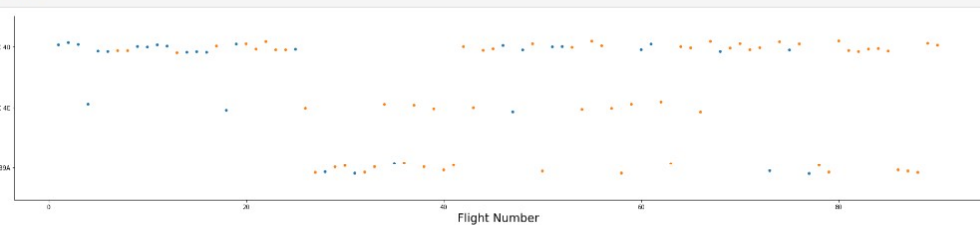
Next, let's drill down to each site visualize its detailed launch records.

EDA with Visualization Results

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'`

```
# 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, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



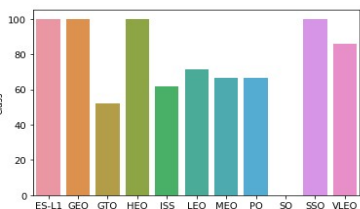
TASK 3: Visualize the relationship between success rate of each orbit type

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

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

```
# HINT use groupby method on Orbit column and get the mean of Class column
temp = df.groupby(["Orbit"]).mean().reset_index()
temp2 = temp[["Orbit", "Class"]]
temp2["Class"] = temp2["Class"]*100
sns.barplot(x = "Orbit", y = "Class", data = temp2)
```

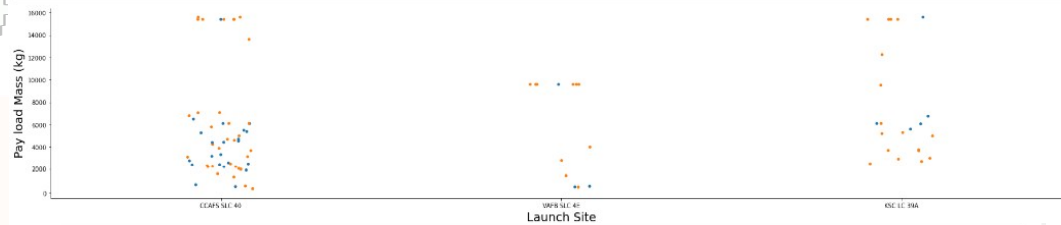
SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using `.loc[row_indexer,col_indexer] = value` instead. See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy



TASK 2: Visualize the relationship between Payload and Launch Site

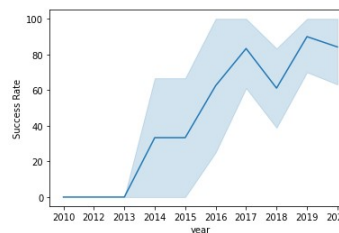
We also want to observe if there is any relationship between launch sites and their payload mass.

```
# 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="PayloadMass", x="LaunchSite", hue="Class", data=df, aspect = 5)
plt.xlabel("Launch Site",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



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

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year = []
df["year"] = Extract_year(year)
df["Success Rate"] = df["Class"] * 100
sns.lineplot(data = df, x = "year", y = "Success Rate")
```



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

EDA with SQL Results

Names of the unique launch sites in the space mission

```
SELECT DISTINCT LAUNCH_SITE FROM SPACEXDATASET
```

```
ibm_db_sa://nx527972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
```

Records where launch sites begin with the string 'CCA'

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

```
ibm_db_sa://nx527972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
```

time_utc	booster_version	launch_site	payload	payload_mass_kg	orbit	customer	mission_outcome	landing_outcome
18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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)
07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
00:25:00	F9 v1.0 B0006	CCAFS LC-40	Dragon V CRS-1	500	LEO	NASA (COTS)	Success	No attempt

Task 3

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

```
%sql select sum(payload_mass_kg_) as sum from SPACEXDATASET where customer like 'NASA (CRS)'
```

```
* ibm_db_sa://nx527972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

SUM

45596

Task 4

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

```
%sql select avg(payload_mass_kg_) as Average from SPACEXDATASET where booster_version like 'F9 v1.1%'
```

```
* ibm_db_sa://nx527972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.
```

average

2534

EDA with SQL Results

the first successful landing outcome in ground pad was achieved.

date) as Date from SPACEXDATASET where mission_outcome like 'Success'

27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

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

er_version from SPACEXDATASET where (mission_outcome like 'Success')

kg BETWEEN 4000 AND 6000) AND (landing_outcome like 'Success (drone ship)')

27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

Activate Windows
Go to Settings to activate Windows

which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015

THNAME(Date) as Month, landing_outcome, booster_version, launch_site

SET where DATE like '2015%' AND landing_outcome like 'Failure (drone ship)'

xs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB

outcome booster_version launch_site

one ship) F9 v1.1 B1012 CCAFS LC-40

one ship) F9 v1.1 B1015 CCAFS LC-40

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT mission_outcome, count(*) as Count FROM SPACEXDATASET GROUP by mission_outcome ORDER BY mission_outcome
```

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

mission_outcome	COUNT
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

```
maxv = %sql select max(payload_mass__kg_) from SPACEXDATASET
```

```
%sql select booster_version from SPACEXDATASET where  
payload_mass__kg_=(select max(payload_mass__kg_) from SPACEXDATASET)
```

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

booster_version

F9 B5 B1048.4

F9 B5 B1049.4

F9 B5 B1051.3

PAGE 1 OF 1

Activate Windows
Go to Settings to activate Windows

Task 10

Rank the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

```
%sql select landing_outcome, count(*) as count from SPACEXDATASET  
where Date >= '2010-06-04' AND Date <= '2017-03-20'  
GROUP by landing_outcome ORDER BY count Desc
```

* ibm_db_sa://nxs27972:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/BLUDB
Done.

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

Developer

SKILLS NETWORK

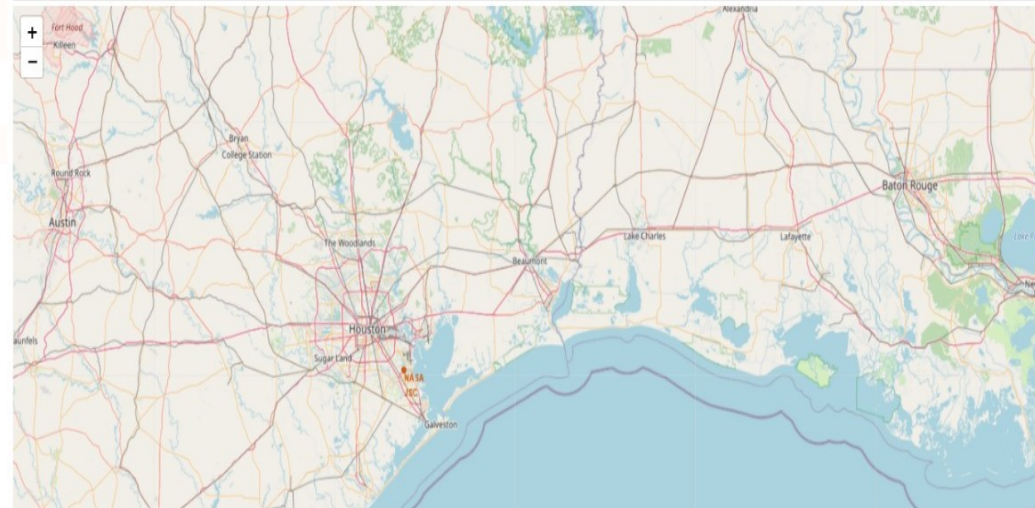
Interactive map with Folium Results

```
nt sub-columns: 'Launch Site', 'Lat(Latitude)', 'Long(Longitude)', 'class'  
ex_df[['Launch Site', 'Lat', 'Long', 'class']]  
= spacex_df.groupby(['Launch Site'], as_index=False).first()  
= launch_sites_df[['Launch Site', 'Lat', 'Long', 'class']]
```

Lat	Long	class
3.562302	-80.577356	0
3.563197	-80.576820	1
3.573255	-80.646895	1
4.632834	-120.610746	0

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

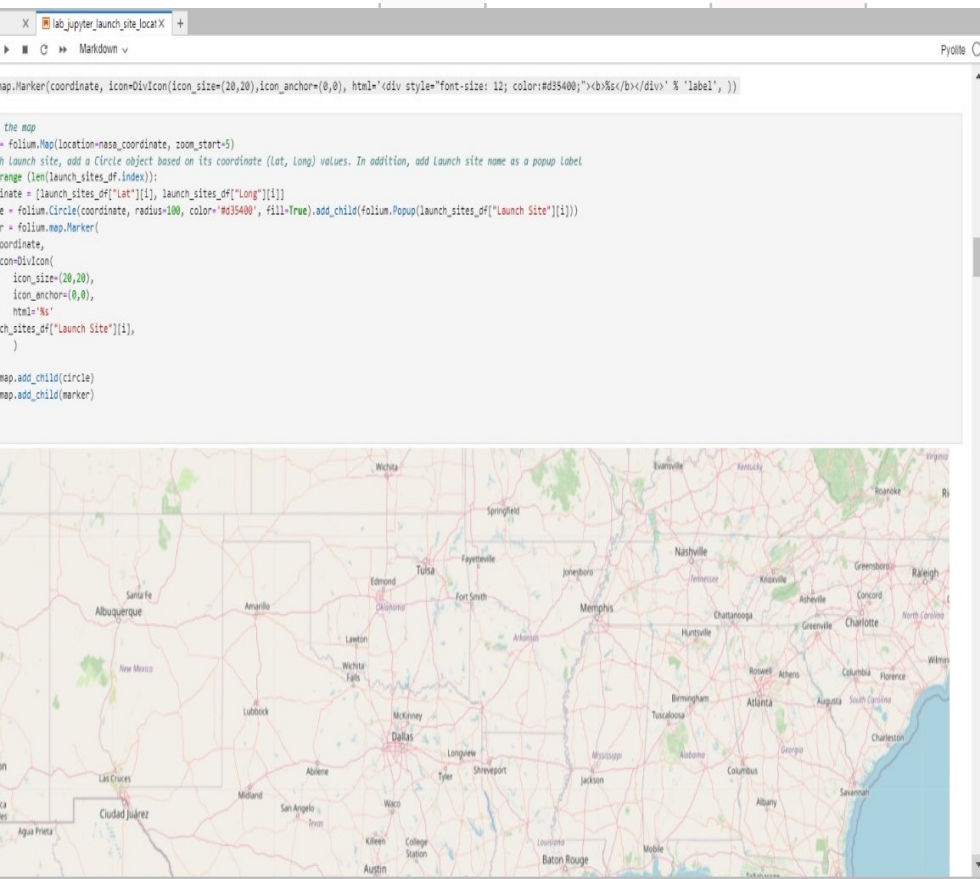
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name  
circle = folium.Circle(nasa_coordinate, radius=1000, color='blue', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))  
# Create a blue circle at NASA Johnson Space Center's coordinate with an 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:blue;"><b>NASA</b></div>' % 'NASA JSC',  
    )  
)  
site_map.add_child(circle)  
site_map.add_child(marker)
```



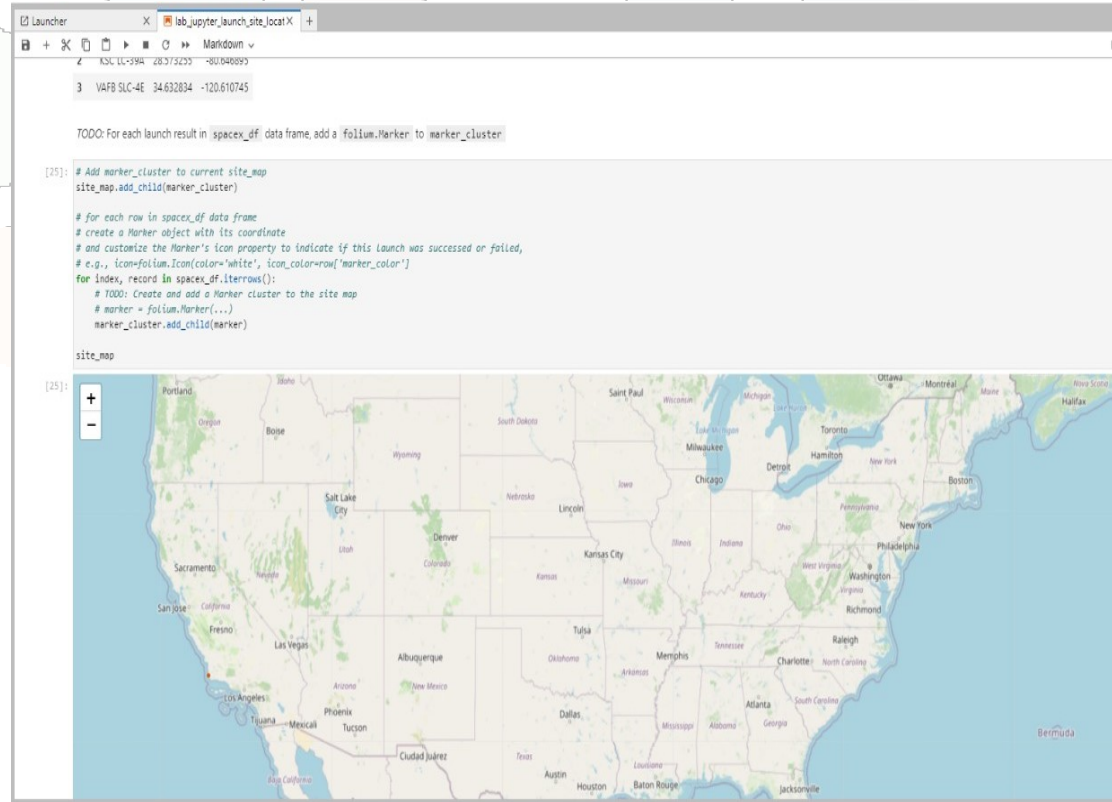
Developer

SKILLS NETWORK

Interactive map with Folium results

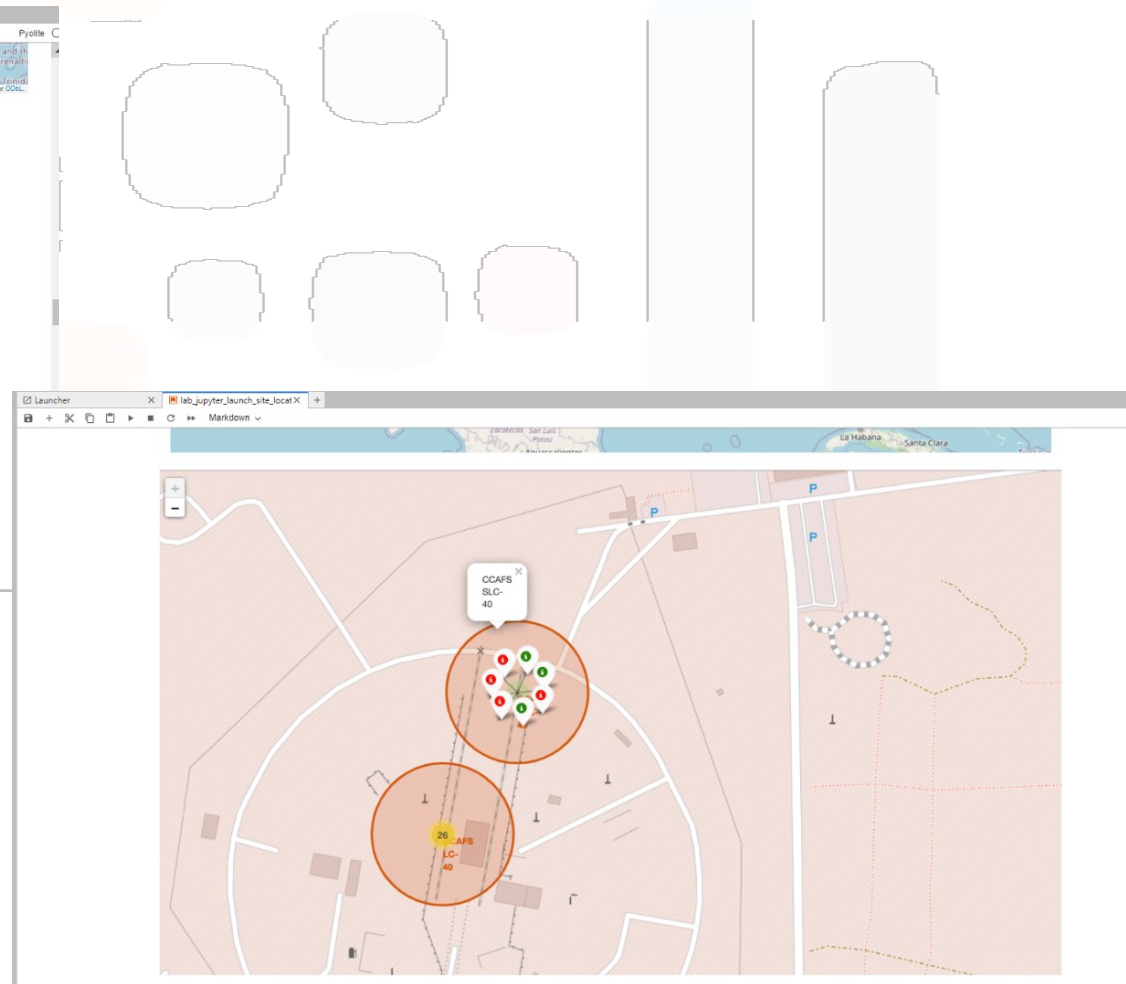
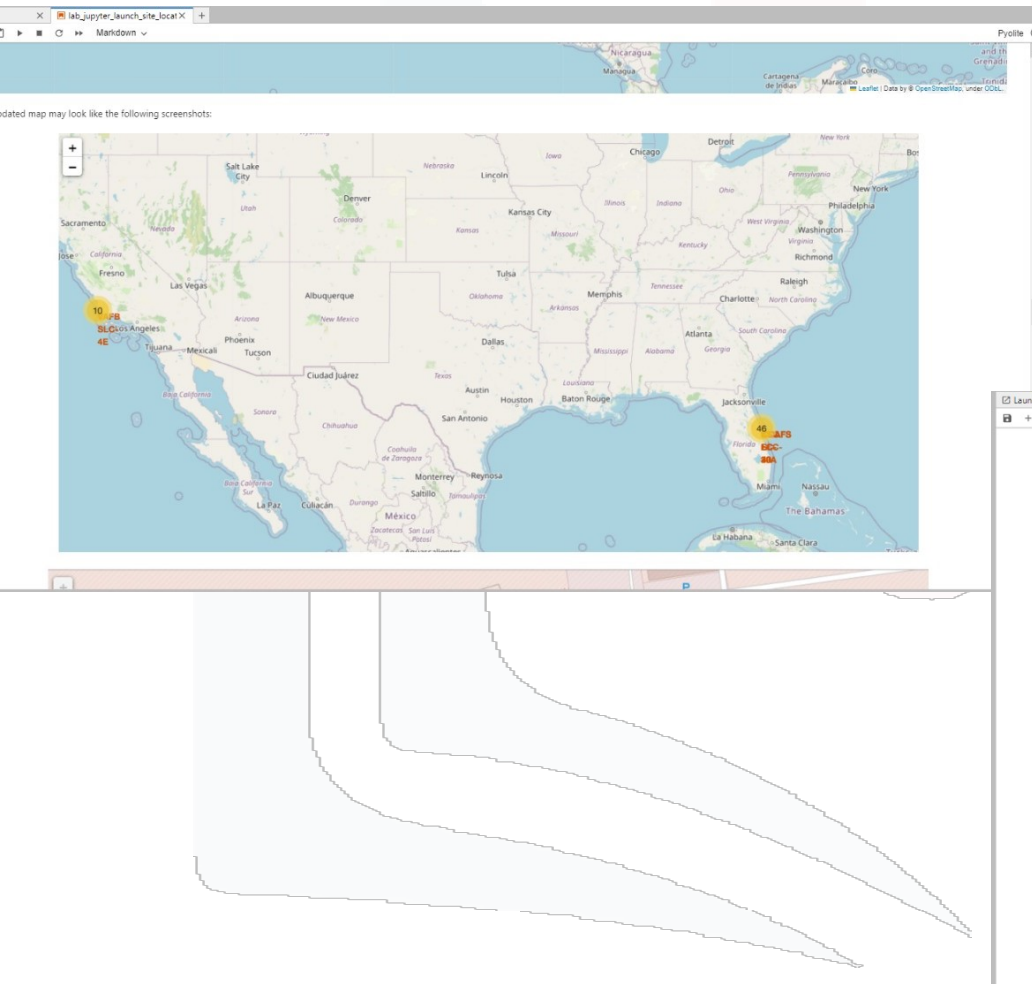


Developer



SKILLS NETWORK

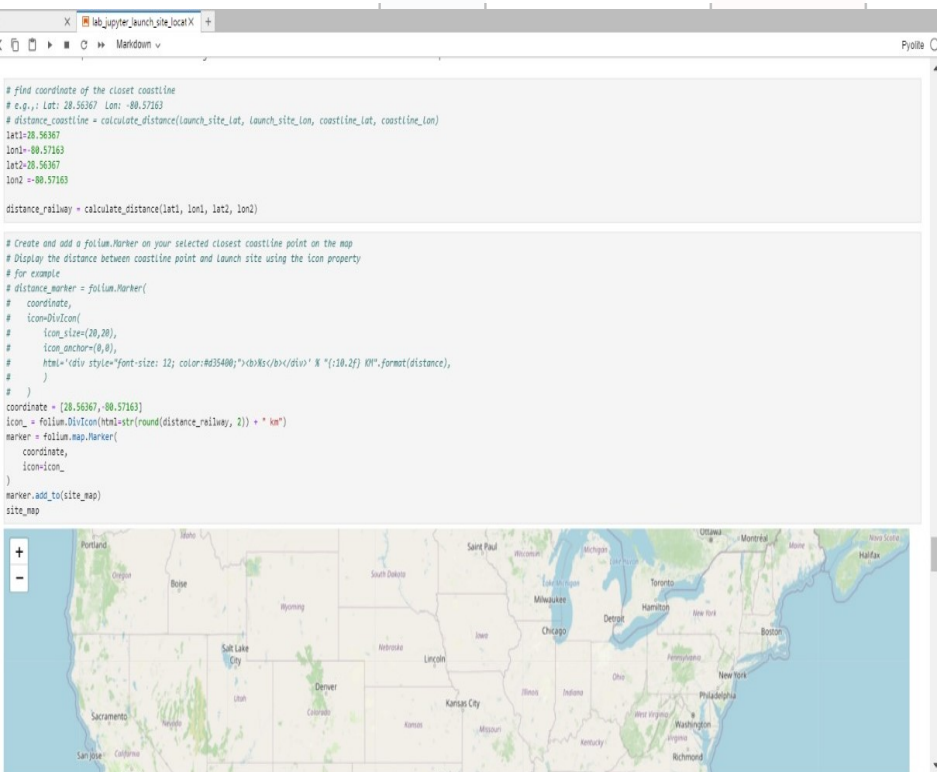
Interactive map with Folium results



Developer

SKILLS NETWORK

Interactive map with Folium results



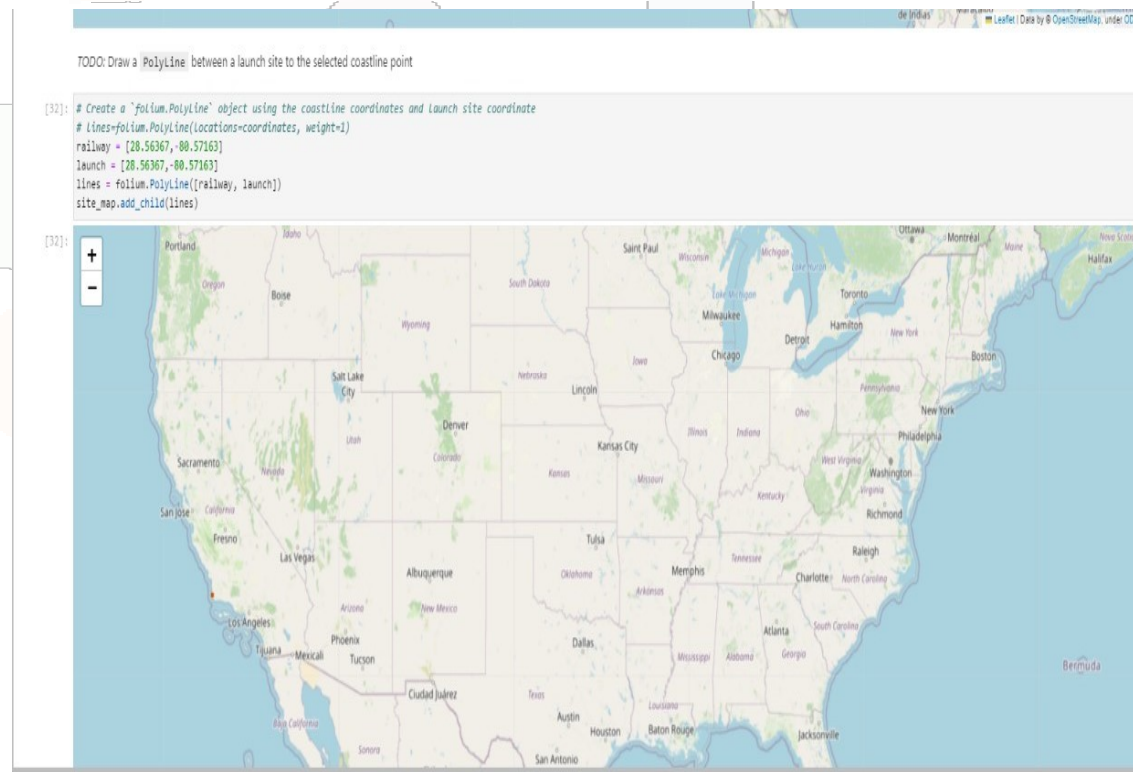
The screenshot shows a Jupyter Notebook window with the following code:

```
# find coordinate of the closest coastline
# e.g., lat: 28.56367 lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
lat1=28.56367
lon1=-80.57163
lat2=28.56367
lon2=-80.57163
distance_railway = calculate_distance(lat1, lon1, lat2, lon2)

# 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
# for example
# distance_marker = folium.Marker(
#     coordinate,
#     icon=DivIcon(
#         icon_size=(20,20),
#         icon_anchor=(0,0),
#         html='<div style="font-size: 12; color:#d35400;"><b>{0}</b></div>' * "{:10.2f} km".format(distance),
#     )
# )
coordinate = [28.56367, -80.57163]
icon = folium.DivIcon(html=str(round(distance_railway, 2)) + " km")
marker = folium.Marker(
    coordinate,
    icon=icon,
)
marker.add_to(site_map)
site_map
```

The map below the code shows the United States with a red dot on the Gulf of Mexico coast, representing the launch site. The map is titled "Pytole" in the top right corner.

Developer



The screenshot shows a Jupyter Notebook window with the following code:

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

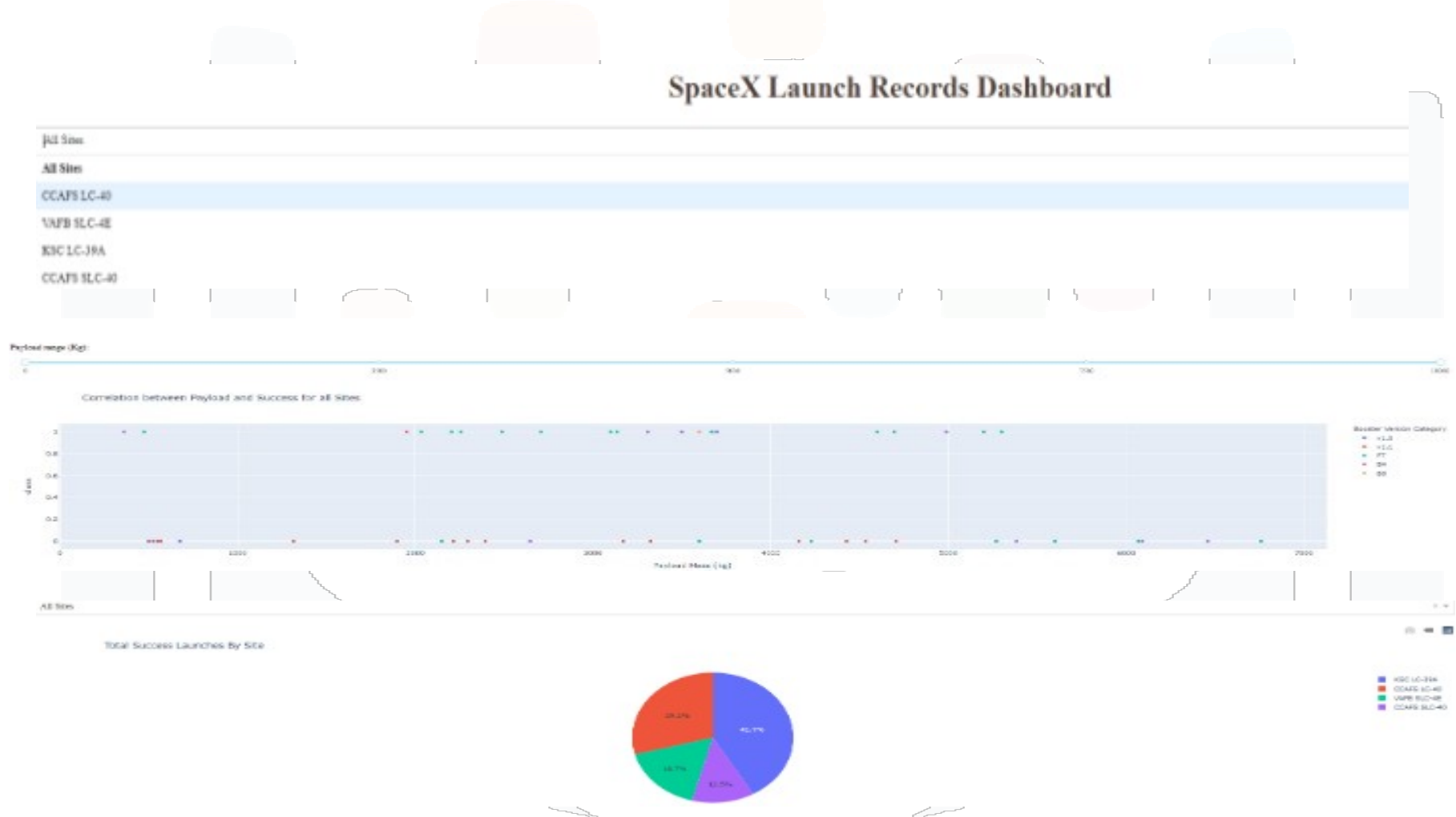
[32]: # Create a "folium.PolyLine" object using the coastline coordinates and launch site coordinate
# Lines=folium.PolyLine(locations=coordinates, weight=1)
railway = [28.56367, -80.57163]
launch = [28.56367, -80.57163]
lines = folium.PolyLine([railway, launch])
site_map.add_child(lines)

[32]:
```

The map below the code shows the United States with a red dot on the Gulf of Mexico coast, representing the launch site. The map is titled "Pytole" in the top right corner.

SKILLS NETWORK

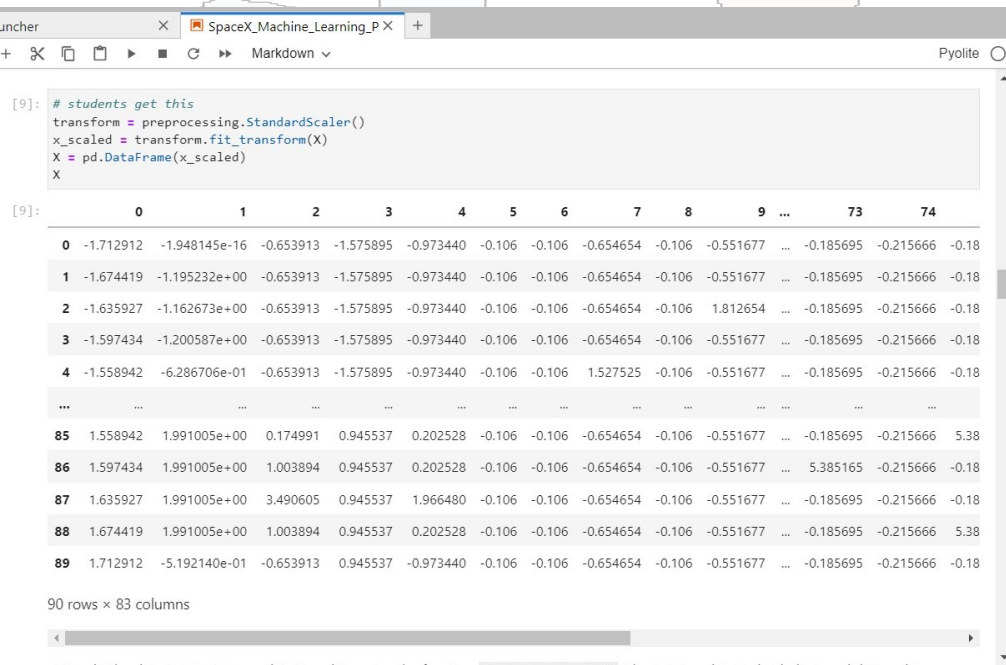
Plotly dashboard results



Developer

SKILLS NETWORK

Predictive Analysis Results



The screenshot shows a Jupyter Notebook interface with a code cell and a table preview. The code cell contains the following Python code:

```
[9]: # students get this
transform = preprocessing.StandardScaler()
x_scaled = transform.fit_transform(X)
X = pd.DataFrame(x_scaled)
X
```

The table preview shows the first 90 rows and 83 columns of the DataFrame. The columns are indexed from 0 to 74, with an ellipsis indicating columns 75 to 79. The rows are indexed from 0 to 89, with an ellipsis indicating rows 90 to 89. The data values are numerical, ranging from approximately -1.712912 to 1.991005e+00.

	0	1	2	3	4	5	6	7	8	9	...	73	74
0	-1.712912	-1.948145e-16	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
1	-1.674419	-1.195232e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
2	-1.635927	-1.162673e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	1.812654	...	-0.185695	-0.215666
3	-1.597434	-1.200587e+00	-0.653913	-1.575895	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
4	-1.558942	-6.286706e-01	-0.653913	-1.575895	-0.973440	-0.106	-0.106	1.527525	-0.106	-0.551677	...	-0.185695	-0.215666
...
85	1.558942	1.991005e+00	0.174991	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
86	1.597434	1.991005e+00	1.003894	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	5.385165	-0.215666
87	1.635927	1.991005e+00	3.490605	0.945537	1.966480	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
88	1.674419	1.991005e+00	1.003894	0.945537	0.202528	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666
89	1.712912	-5.192140e-01	-0.653913	0.945537	-0.973440	-0.106	-0.106	-0.654654	-0.106	-0.551677	...	-0.185695	-0.215666

90 rows x 83 columns

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=2`. The training data and test data should be assigned to the following labels.

```
X_train, X_test, Y_train, Y_test
```

```
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.

```
Y_test.shape
```

```
(18,)
```

Predictive Analysis Results

4

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

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

```
logreg = LogisticRegression()  
logreg.fit(X_train, Y_train)  
logreg_cv = GridSearchCV(logreg, parameters, cv = 10)  
logreg_cv.fit(X_train, Y_train)
```

```
logreg_cv.best_estimator_ = LogisticRegression()  
logreg_cv.best_params_ = {'C': 0.01, 'penalty': 'l2',  
                           'solver': 'lbfgs'}
```

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

```
print('Best parameters: ', logreg_cv.best_params_)  
print('Accuracy: ', logreg_cv.best_score_)
```

```
Best parameters: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}  
Accuracy: 0.8464285714285713
```

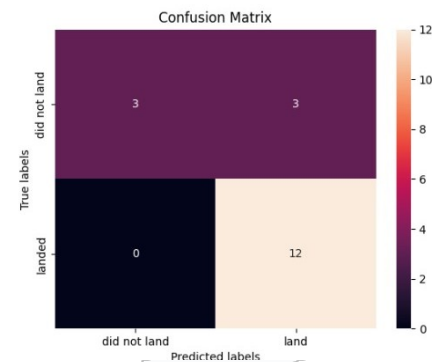
Calculate the accuracy on the test data using the method `score`:

```
[15]: accu = []  
      methods = []  
      accu.append(logreg_cv.score(X_test, Y_test))  
      methods.append('logistic regression')  
      logreg_cv.score(X_test, Y_test)
```

```
[15]: 0.8333333333333334
```

Lets look at the confusion matrix:

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



Developer

SKILLS NETWORK

Predictive Analysis Results

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`.

```
kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid'),  
'C': np.logspace(-3, 3, 5),  
'gamma': np.logspace(-3, 3, 5)}
```

```
GridSearchCV(svm, parameters, cv = 10)  
fit(X_train, Y_train)
```

```
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')})
```

```
best_params_ : (best parameters) ", svm_cv.best_params_  
best_score_ : ", svm_cv.best_score_
```

```
best_params_ : (best parameters) {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}  
2142857142856
```

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

```
svm_cv.score(X_test, Y_test)  
support vector machine'  
svm_cv.score(X_test, Y_test)
```

0.8333333333333334

We can plot the confusion matrix



TASK 7

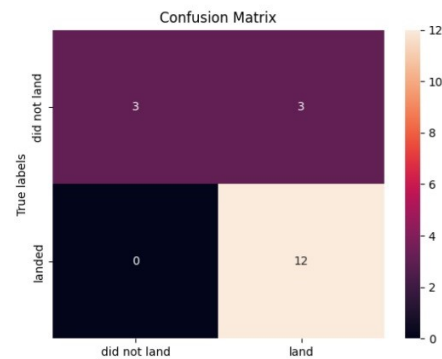
Calculate the accuracy on the test data using the method `score`:

```
accu.append(svm_cv.score(X_test, Y_test))  
methods.append('support vector machine')  
svm_cv.score(X_test, Y_test)
```

0.8333333333333334

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)  
plot_confusion_matrix(Y_test, yhat)  
plt.show()
```



Developer

SKILLS NETWORK

Predictive Analysis Results

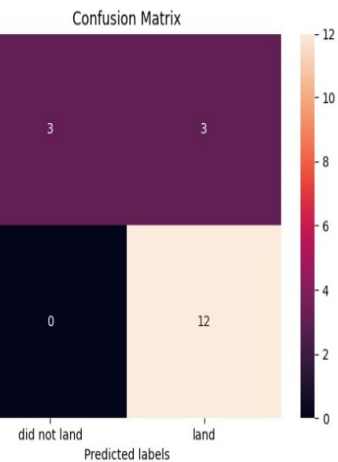
accuracy of tree_cv on the test data using the method score:

```
tree_cv.score(X_test,Y_test))
'decision tree classifier')
tree_cv.score(X_test,Y_test)
```

144

Confusion matrix

```
tree_cv.predict(X_test)
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.

```
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()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv = 10)
knn_cv.fit(X_train, Y_train)
```

```
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]})
```

```
print("tuned hyperparameters :(best parameters) ", knn_cv.best_params_)
print("accuracy :", knn_cv.best_score_)
```

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

Predictive Analysis Results

TASK 12

Find the method performs best:

```
print(methods)
print(accuracies)
```

```
['logistic regression', 'support vector machine', 'decision tree classifier', 'k nearest neighbors']
[0.8333333333333334, 0.8333333333333334, 0.9444444444444444, 0.8333333333333334]
```

Authors

Developer

SKILLS NETWORK

Conclusion

Thus the models have been built using the SpaceX data after performing exploratory data analysis and also visualizing the transformed data. The machine learning models were built(classification) algorithms such as Logistic regression, support vector machine, decision tree classifier and k-NN algorithm and we obtained an accuracy of 83.34%.i.e., the launch of the first stage of the spaceX falcon9.