IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tim Lee
9 August 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using SpaceX API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Map with Folium
  - Dash Board with Plotly Dash
- Summary of all results
  - EDA Insights
  - Launch Site Analytics and Dashboards
  - Predictive Analysis

Introduction

- ## Project background and context

- 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.

- ## Problems you want to find answers

- 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.

Section 1

Methodology

Methodology

Executive Summary

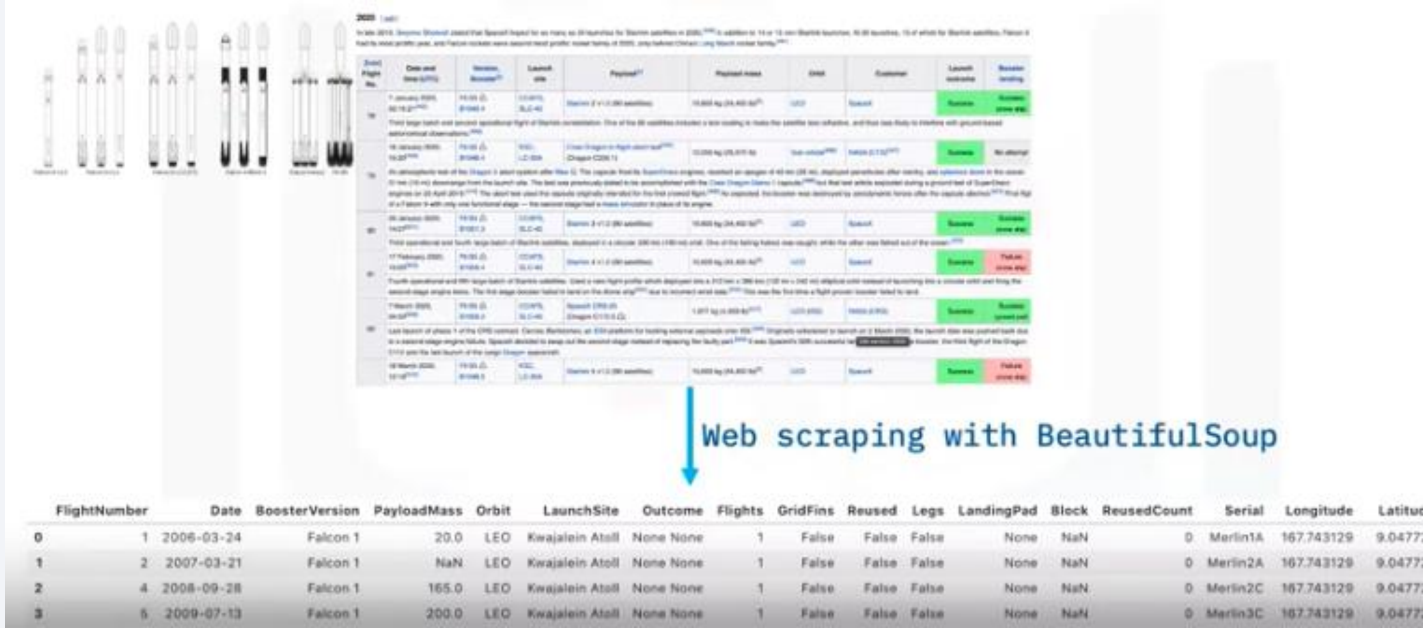
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using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

Data Collection

Data was collected the Data with 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.

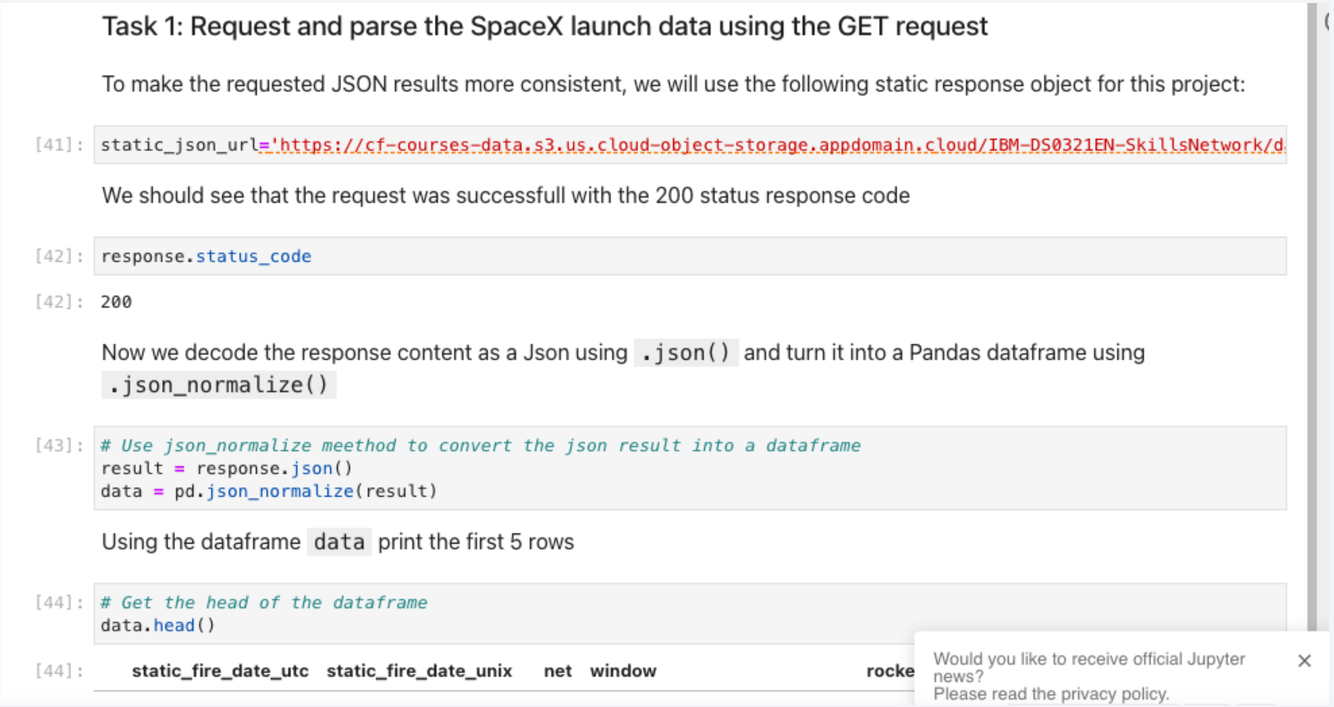
Another popular data source for obtaining Falcon 9 Launch data is web scraping related Wiki pages. Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records.



Data Collection – SpaceX API

- Data was collected using SpaceX API by using a get request to the SpaceX API then requesting and parsing the SpaceX launch data using the GET request, and decoding the response content as a Json result, and then converting into a Pandas data frame

- GitHub URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/c80ff074a064628bac06bb62d94a120037bbb63a/jupyter-labs-spacex-data-collection-api.v2ipynb



Data Collection - Scraping

- Used web scraping to collect launch records from Wikipedia using BeautifulSoup, to extract the launch records from HTML table of the Wikipedia page, and then created a data frame by parsing.

- GitHub URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/c80ff074a064628bac06bb62d94a120037bbb63a/jupyter-labs-webscraping.ipynb

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.

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

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

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

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

```
# Use soup.title attribute
soup.title
```

```
<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

Data Wrangling

- Some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training sup ervised models.
- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attem pted but failed due to an accident;
- Mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.
- GitHub URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/c80ff074a064628bac06bb62d94a120037bbb63a/jupyter-labs-webscraping.ipynb

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`:

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

df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df.head()
#df['Class'].value_counts()
```

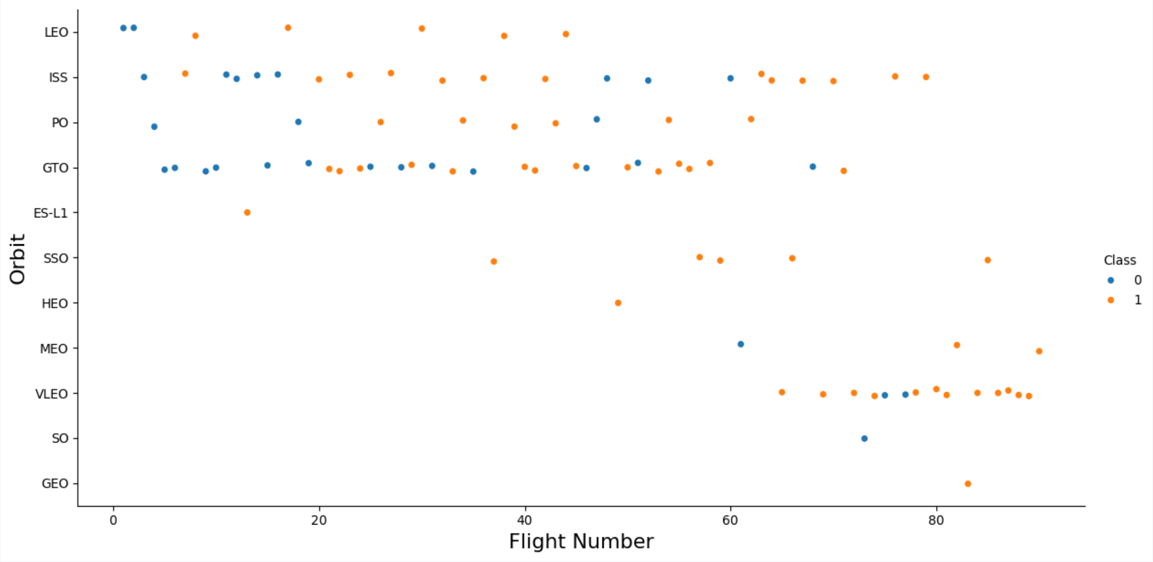
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPac
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN

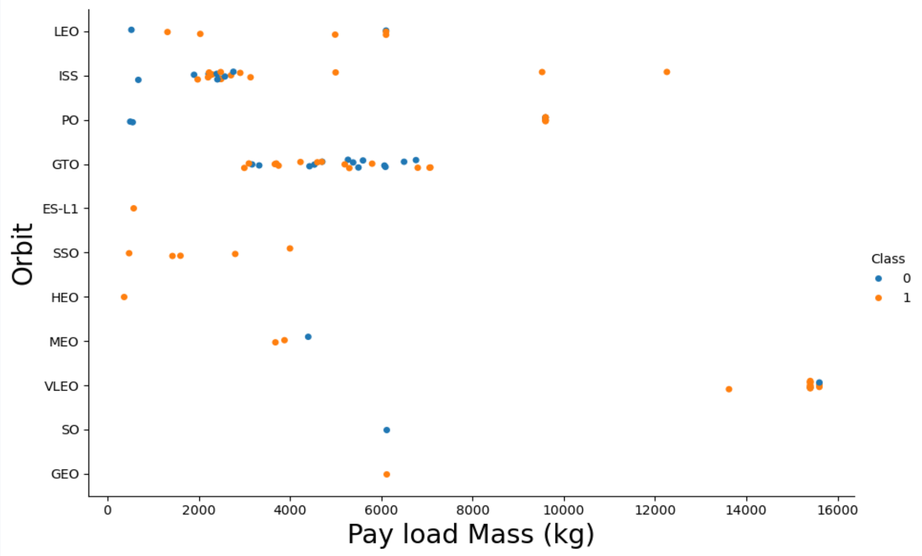
EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with Data Visualization

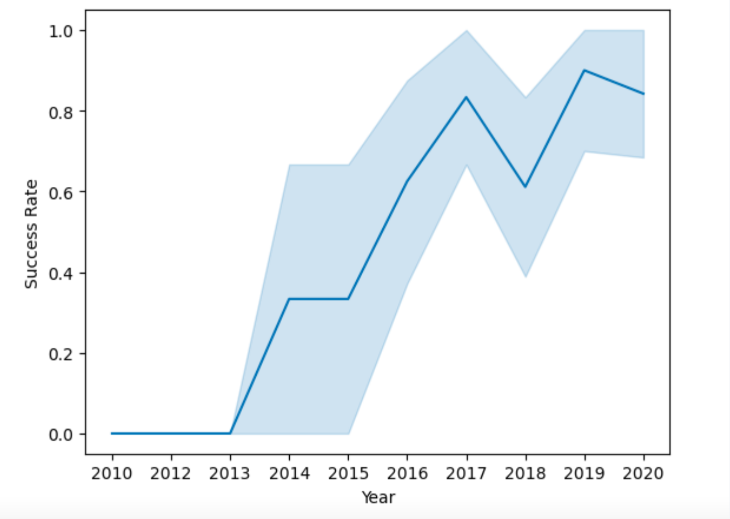
- Scatter plots to visualize the relationship between Flight Number vs Orbit and Payload Mass vs Orbit Payload
- URL: URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/c80ff074a064628bac06bb62d94a120037bbb63a/edadataviz.ipynb

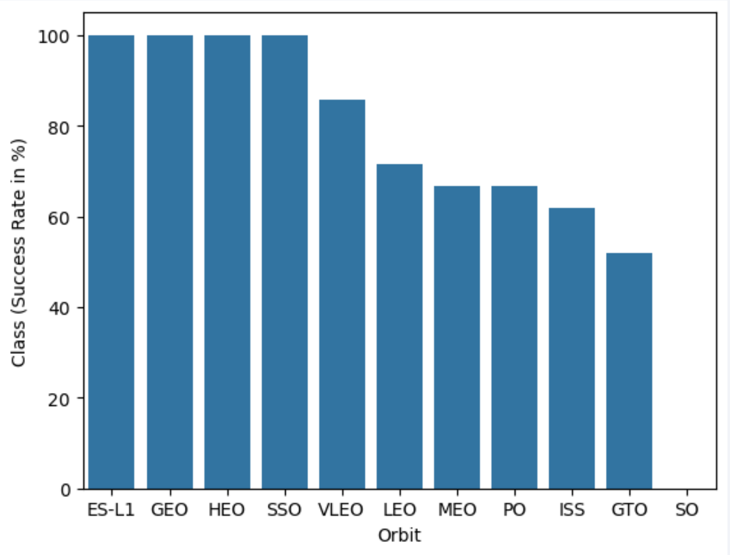




EDA with Data Visualization Continuted

- Bar chart to visualize the relationship between orbit and success rate
- Line plot to visualize the success rate yearly trend.
- URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/c80ff074a064628bac06bb62d94a120037bbb63a/edadataviz.ipynb





EDA with SQL

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display 5 records where launch sites begin with the string 'CCA'
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery¶
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.¶
- 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.
- URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/51d2a66de41ab497bdcbb6978d56b9b90f1353e6/jupyter-labs-eda-sql-coursera_sqllite.ipynb

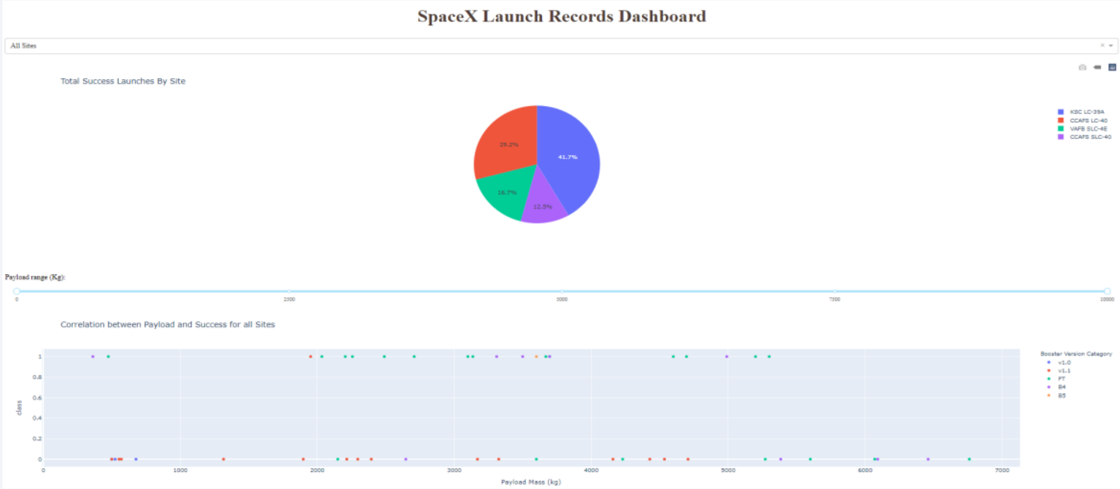
Build an Interactive Map with Folium

- **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
- Now, let's add a circle for each launch site in data frame launch_sites
- *Marker's icon property to indicate if this launch was successed or failed, # e.g., icon=folium.Icon(color='white', icon_color=row['marker_*

- URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/51d2a66de41ab497bdcbb6978d56b9b90f1353e6/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:
    - Adding a Launch Site Drop-down Input Component
    - Adding a callback function to render pie chart based on selected site dropdown
    - Adding a Range Slider to Select Payload
    - Adding a callback function to render the scatter plot
- Pie Chart to visualize the successes between launch sites
- Scatter Plot to visualize the payload vs success rate
- URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/51d2a66de41ab497bdcbb6978d56b9b90f1353e6/spacex_dash_app%20(1).py



Predictive Analysis (Classification)

- 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

- You need present your model development process using key phrases and flowchart
- URL: https://github.com/tlee1112/Coursera-Project-SpaceX/blob/51d2a66de41ab497bdcbb6978d56b9b90f1353e6/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Predictive Analysis (Classification)

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']).

```
Y = data['Class'].to_numpy()
Y.dtype
```

```
dtype('int64')
```

TASK 2

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

```
# students get this
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
X
```

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`

```
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,)
```

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`.

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

TASK 5

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

```
logreg_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

Lets look at the confusion matrix:

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

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`.

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

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

```
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+0
1,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

Results

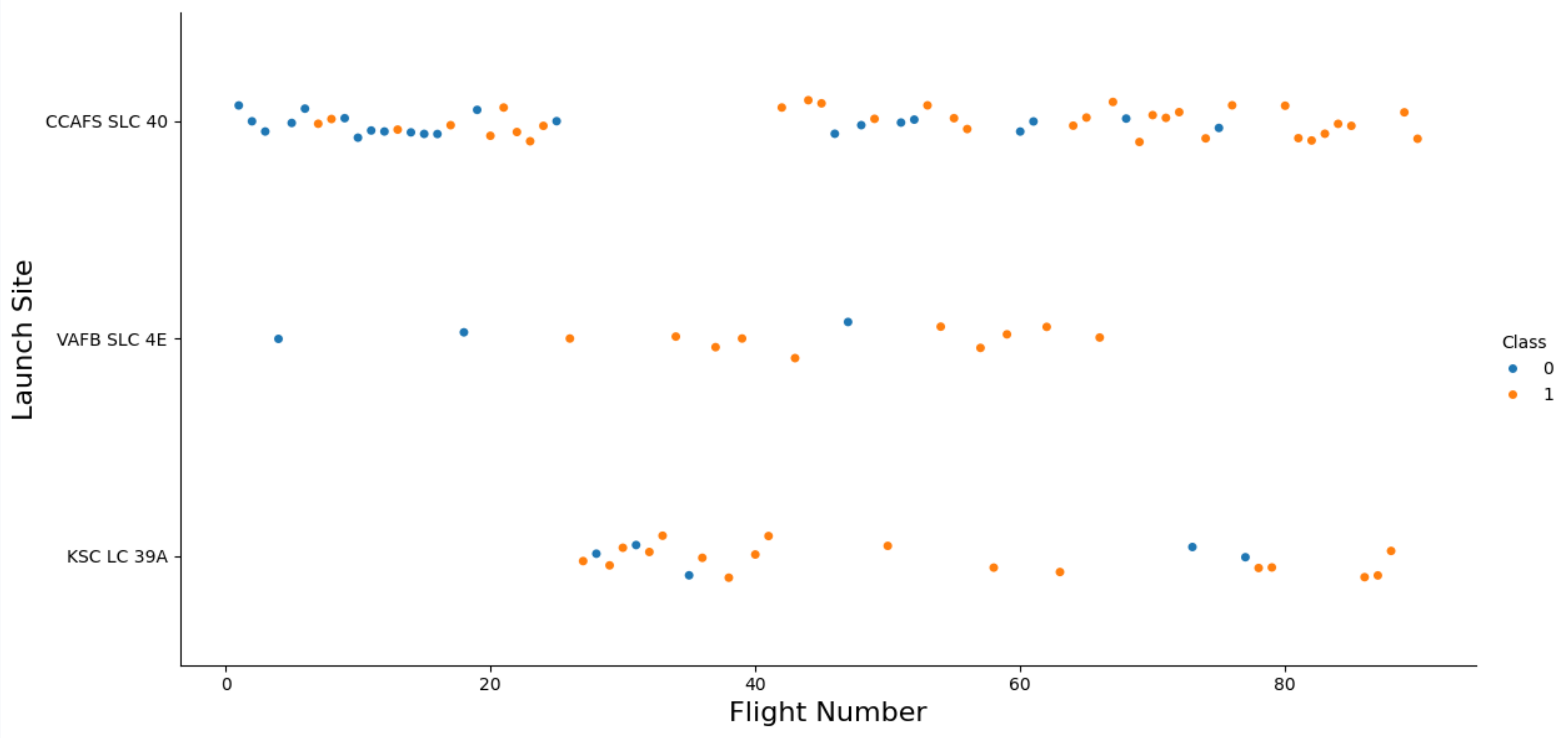
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

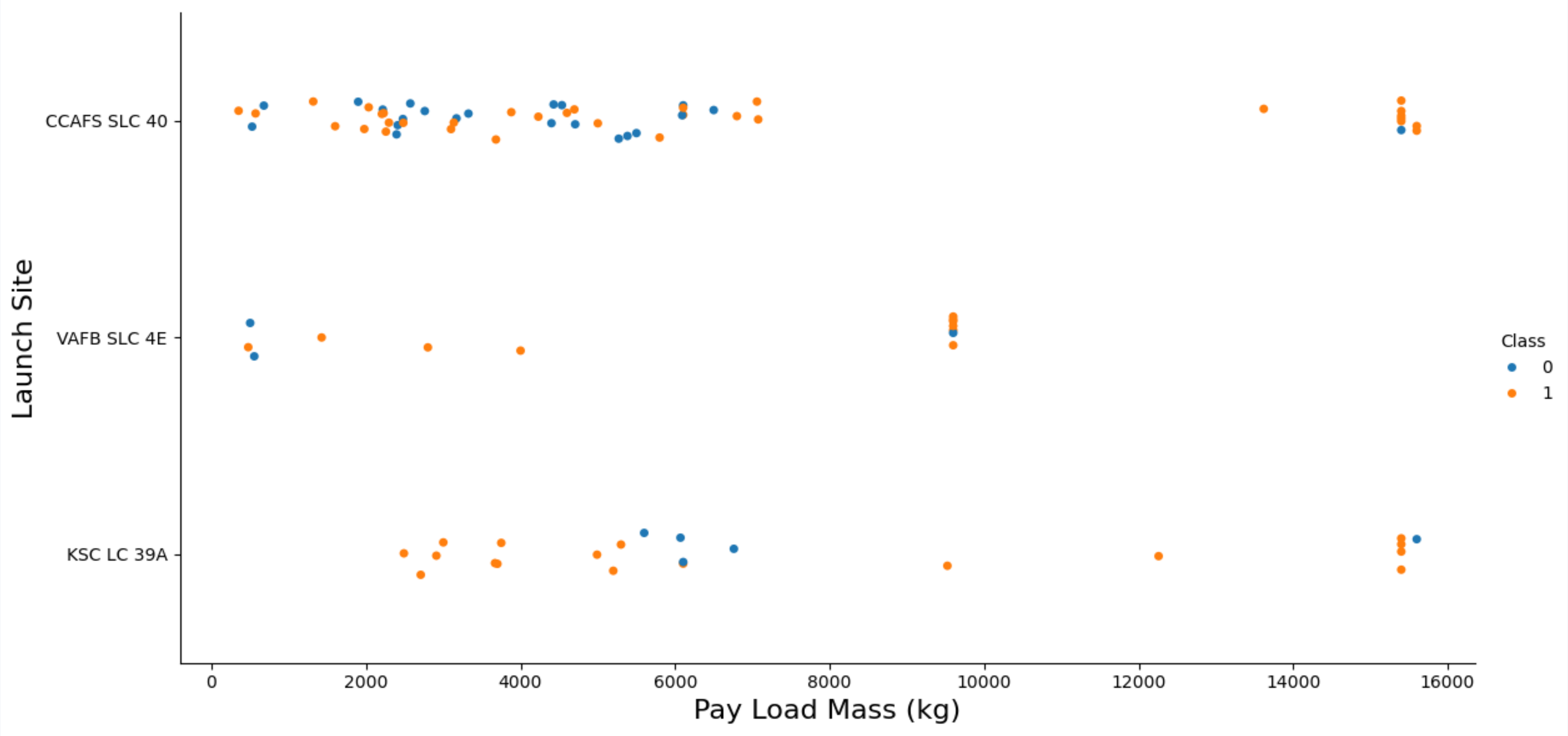
Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site
  - Increasing in success rate as time passes



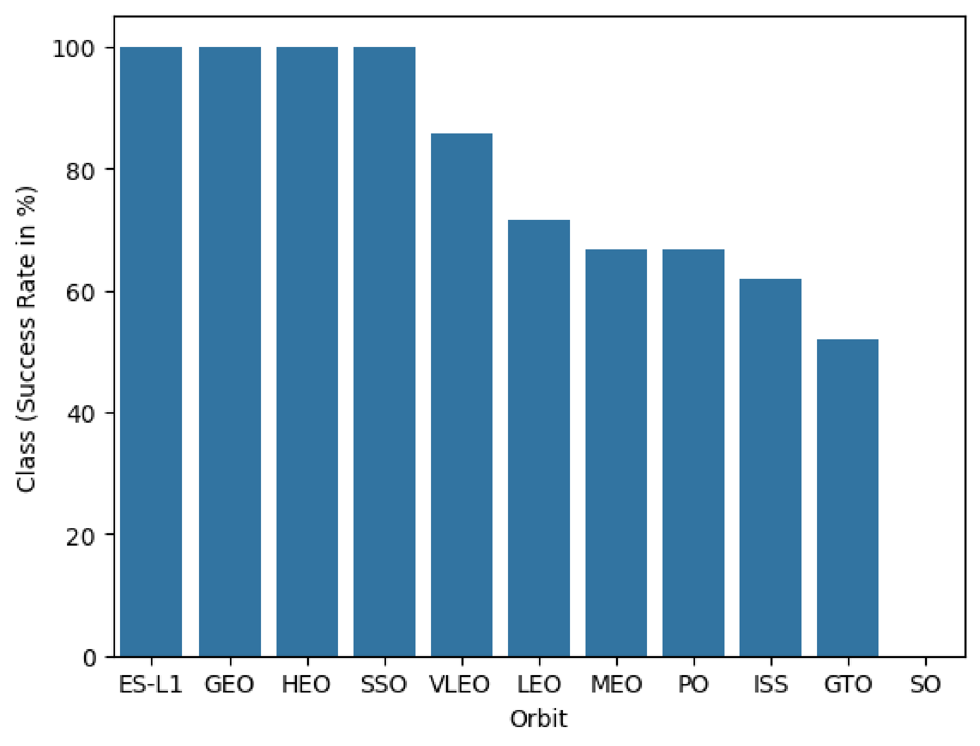
Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site
  - Greater success rate in the higher payload launches



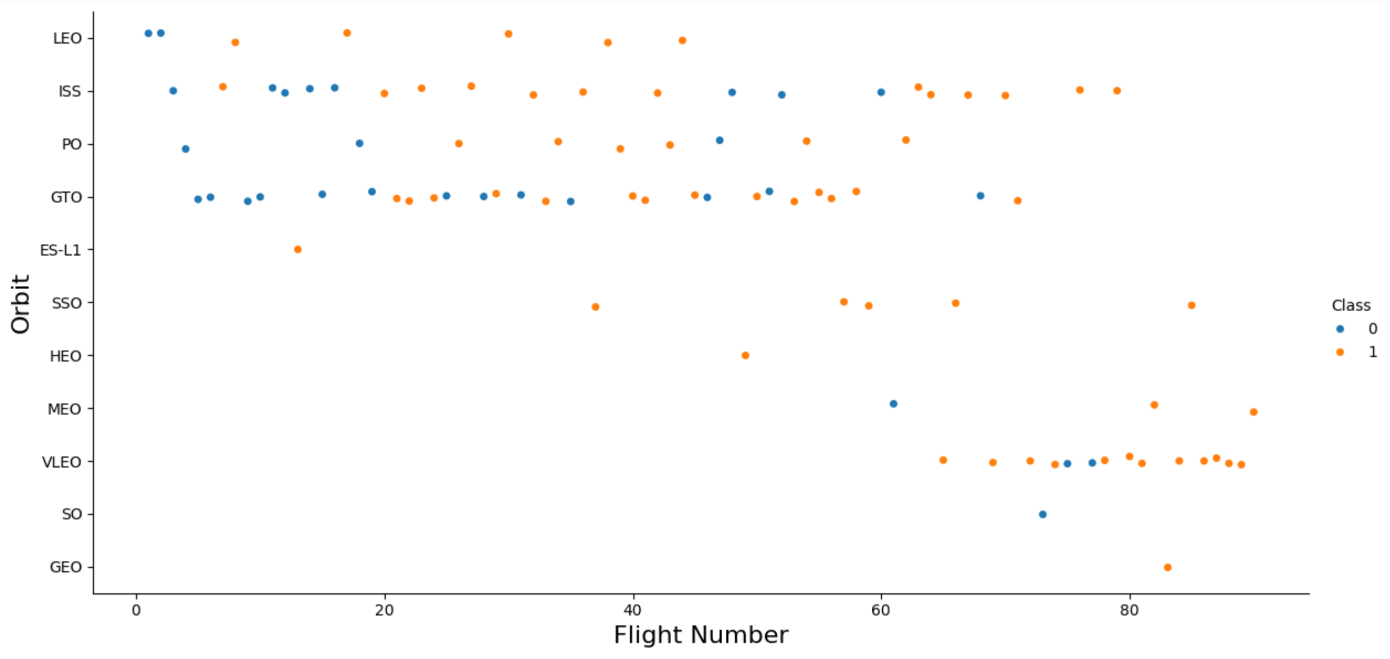
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type
  - ES-L1, GEO, HEO ,SSO have the highest success rate



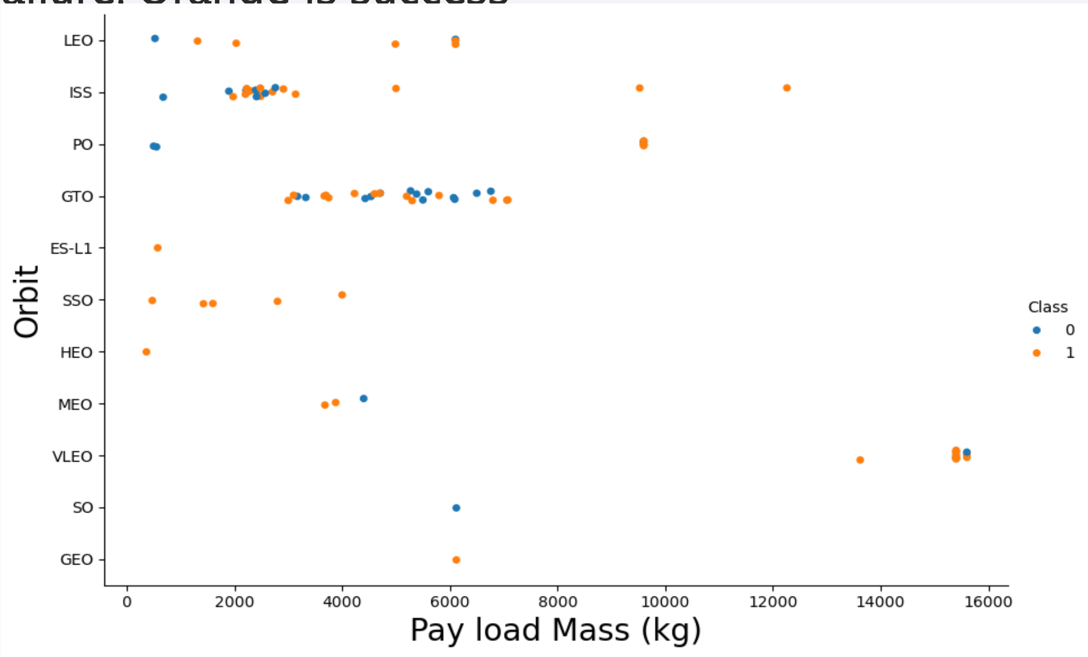
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
  - GTO, PO, ISS, LTO have more early lunches whereas VLEO has the more latest ones
  - More successes in latest years



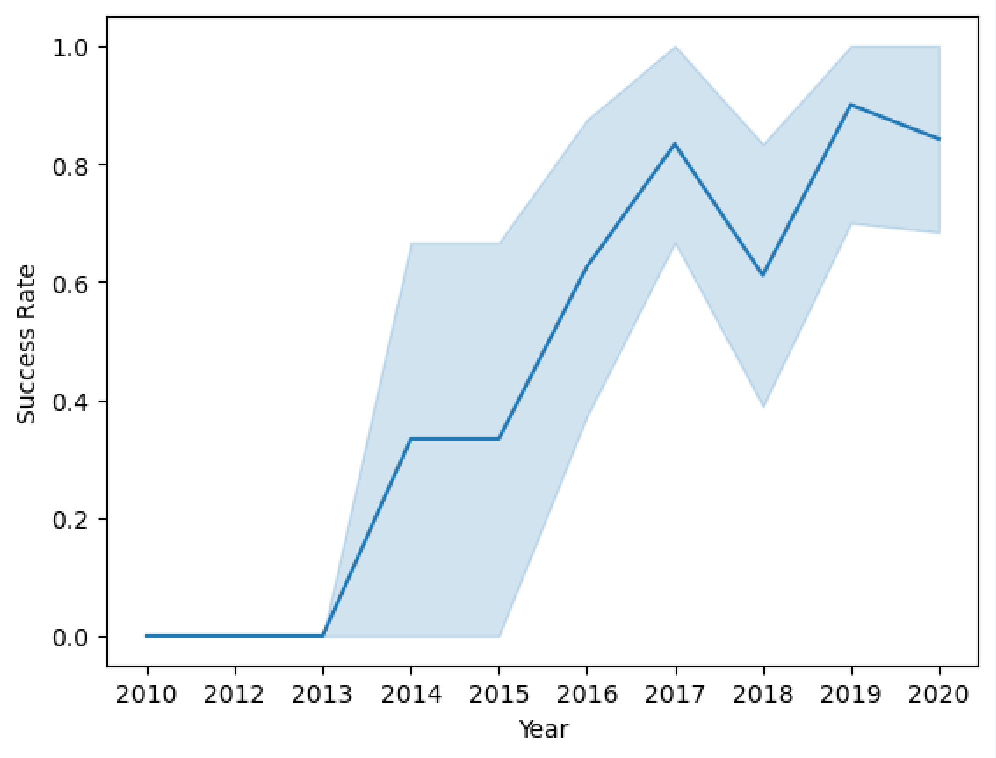
Payload vs. Orbit Type

- Scatterplot of payload vs. orbit type
  - VLEO has high payload mass
  - GTO and ISS have a high number of launches
  - Blue is failure. Orange is success



Launch Success Yearly Trend

- Line chart of yearly average success rate
    - General increase in success rate as years past



All Launch Site Names

- Find the names of the unique launch sites
  - 4 unique names in the query below

Task 1

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

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
  - 5 records with Launch_Site starting with "CCA" queried below

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

```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
  - Total payload mass from NASA (CRS) is 45596 kgs

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass (Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NA
```

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

Total Payload Mass (Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Total payload mass from booster version F9 v1.1 is 2534 kgs

Task 4

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

```
]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass (Kgs)", Booster_Version FROM 'SPACEXTBL' WHERE Booster_Versic
```

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

]:

Payload Mass (Kgs)	Booster_Version
2534.6666666666665	F9 v1.1 B1003

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
  - 1st success on ground pad was in 2015

Task 5

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

Hint:Use min function

```
]: %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (ground pad)";
```

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

]:

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
  - Names shown below

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

```
%sql SELECT DISTINCT Booster_Version, Payload, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE "Landing_Outcome" = "Succes
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 FT B1022	JCSAT-14	4696
F9 FT B1026	JCSAT-16	4600
F9 FT B1021.2	SES-10	5300
F9 FT B1031.2	SES-11 / EchoStar 105	5200

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
  - Table below shows a high number of success

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
  - Table below shows all the boosters that have the max payload of 15600

Task 8

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

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
    - 2 launches in the table below

Task 9

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

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%sql SELECT substr(Date,6,2), substr(Date, 0, 5),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_",
```

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

substr(Date,6,2)	substr(Date, 0, 5)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing_Outcon
01	2015	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone shi
04	2015	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone shi

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order
  - Table below ranks count of landing outcomes

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.

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '2010-06-04' AND '2017-03-
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Lan
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	S
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	S
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	
2016-04-	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	

Section 3

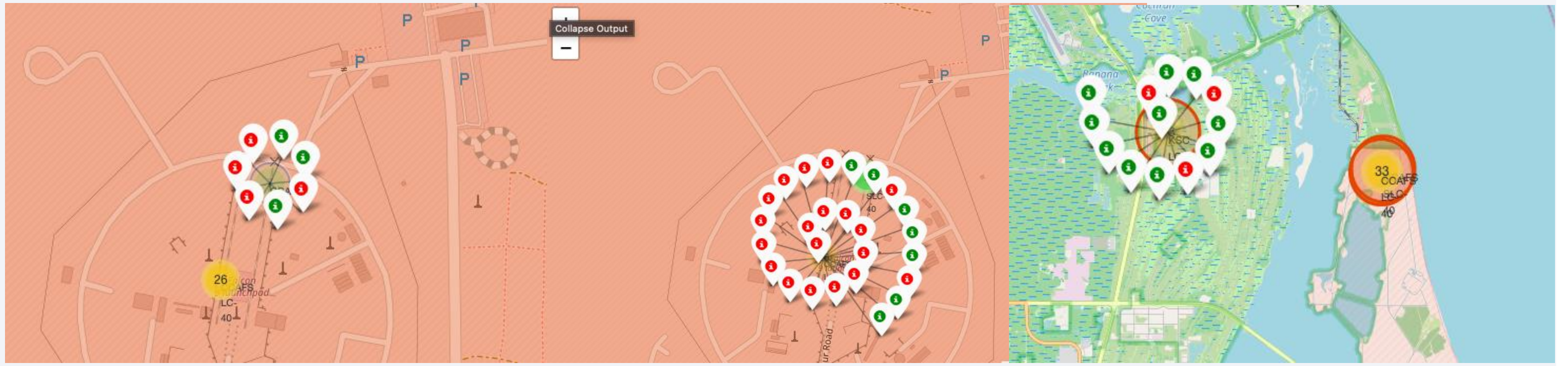
Launch Sites Proximities Analysis

Folium Map – Markers of all Launch Sites on Global Map

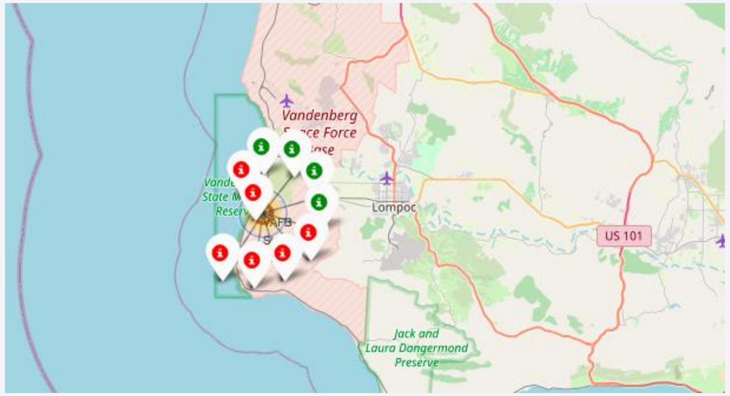
- All launches have been on the coast in CA and FL

Markers at Launch Site

- Screenshots show the successes (green) and failures (red) at each site
- You can see that which sites at a higher success rate and which had lower ones



Florida Launch Sites



California Launch Sites

Launch Site to Coast Distance

- Launch site is approximately .90 km to coastline
  - All launch sites are nearby coastlines

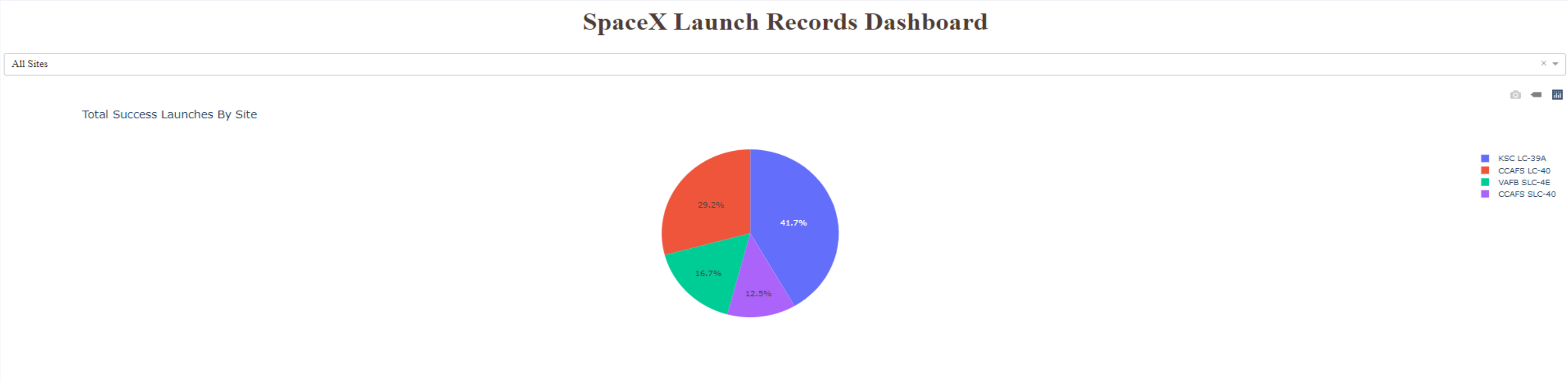
Section 4

Build a Dashboard with Plotly Dash

Launch Site Successes

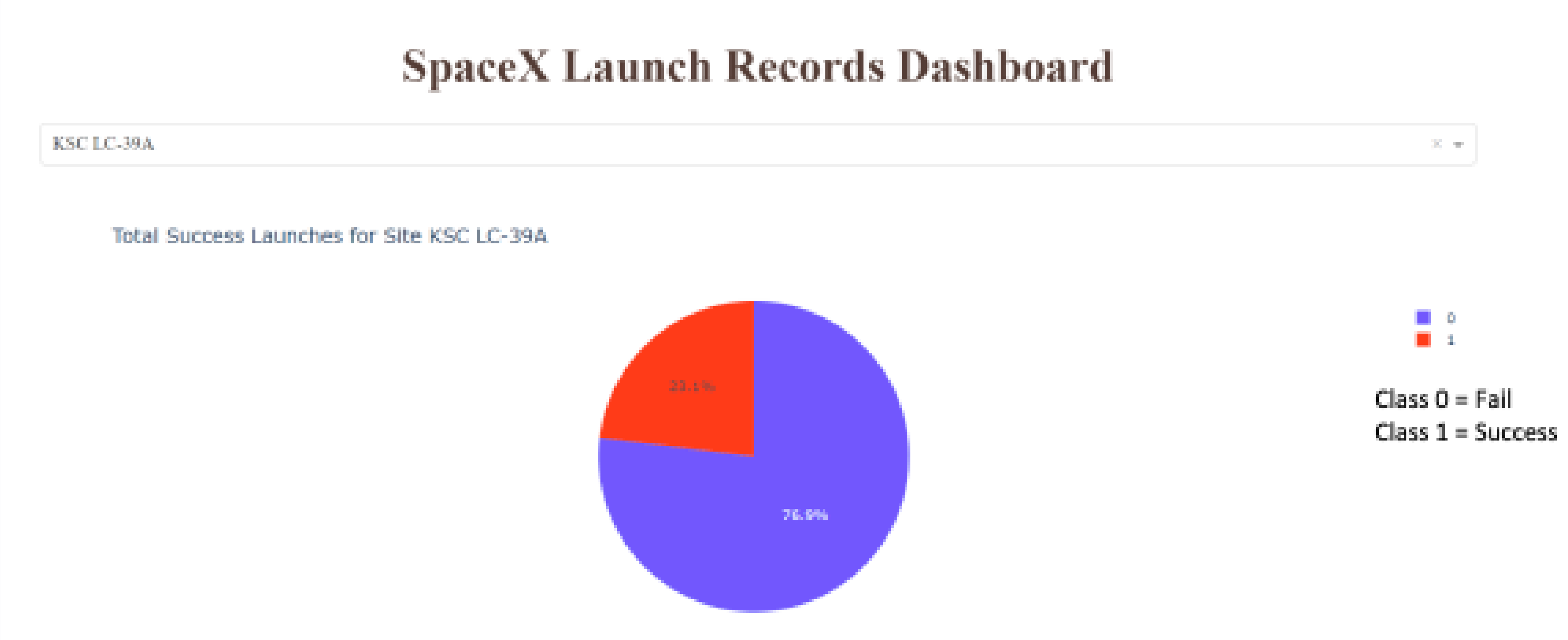
- Launch success count for all sites, in a piechart

- KSC LC-39A has the highest launch success rate at 41.7% and launch site CCAFS SLC-40 has the lowest success rate of 12.5%



KSC LC-39A Launch Success vs Failure

- Pie Chart Shows KSC LC-39A launch site had roughly a 77% success rate



Payload vs Success Rate

- Scatterplot shows payloads in the 2k to 5k kgs range have the highest success rate

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart
- All models had similar Accuracy

TASK 12

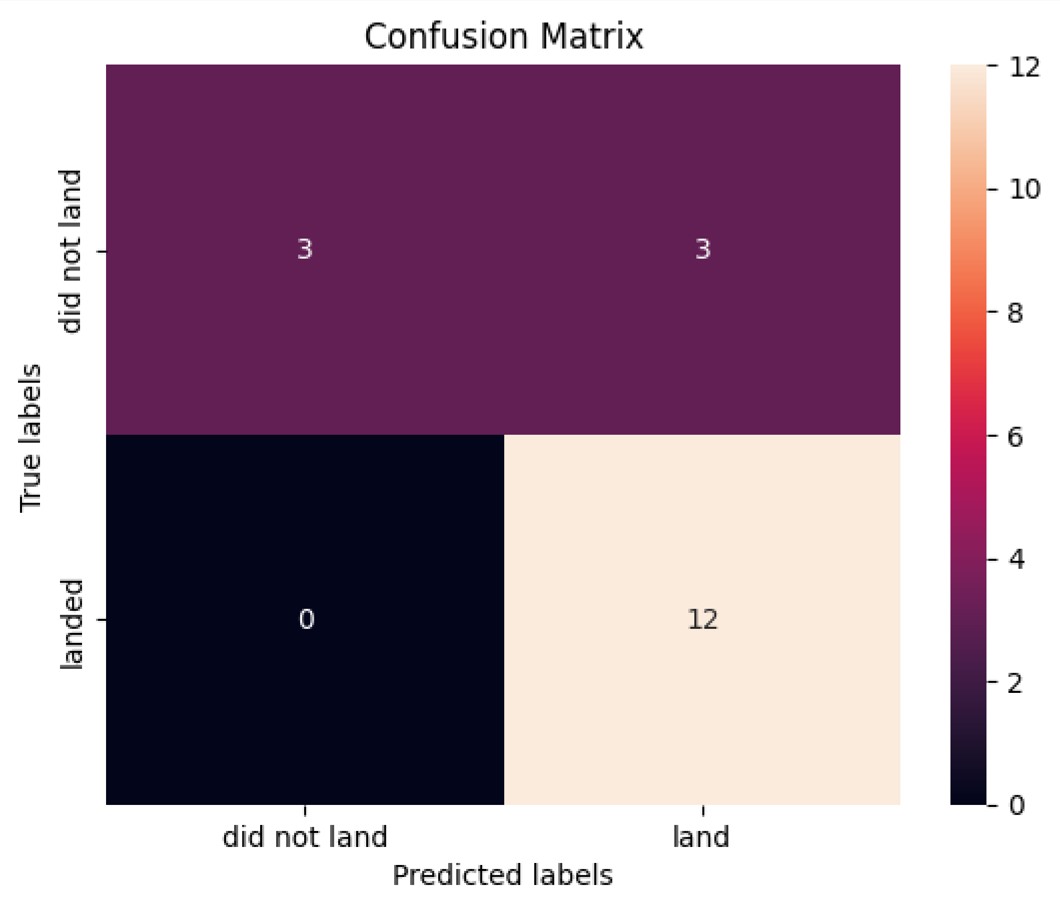
Find the method performs best:

```
Results = pd.DataFrame({'Method': ['Test Data Accuracy']})
KNN_Accuracy=knn_cv.score(X_test, Y_test)
Decision_Tree_Accuracy=tree_cv.score(X_test, Y_test)
SVM_Accuracy=svm_cv.score(X_test, Y_test)
Logistic_Regression=logreg_cv.score(X_test, Y_test)
Results['Logistic_Reg'] = [Logistic_Regression]
Results['SVM'] = [SVM_Accuracy]
Results['Decision Tree'] = [Decision_Tree_Accuracy]
Results['KNN'] = [KNN_Accuracy]
Results.transpose()
```

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.888889
KNN	0.833333

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation
- Confusion Matrix Outputs:
    - 12 True positive
    - 3 True negative
    - 3 False positive
    - 0 False Negative



Conclusions

- Equator: All the launch sites are near the equator
- Coast: All the launch sites are close to the coast
- Launch Success: Success rate increased over time
- KSC LC-39A: Has the highest success rate for launch site
- Orbits: ES-L1, GEO, HEO, and SSO have a higher success rate
- Payload Mass: The higher the payload mass, the higher the success rate
- All model accuracies were similar

Appendix

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

Thank you!