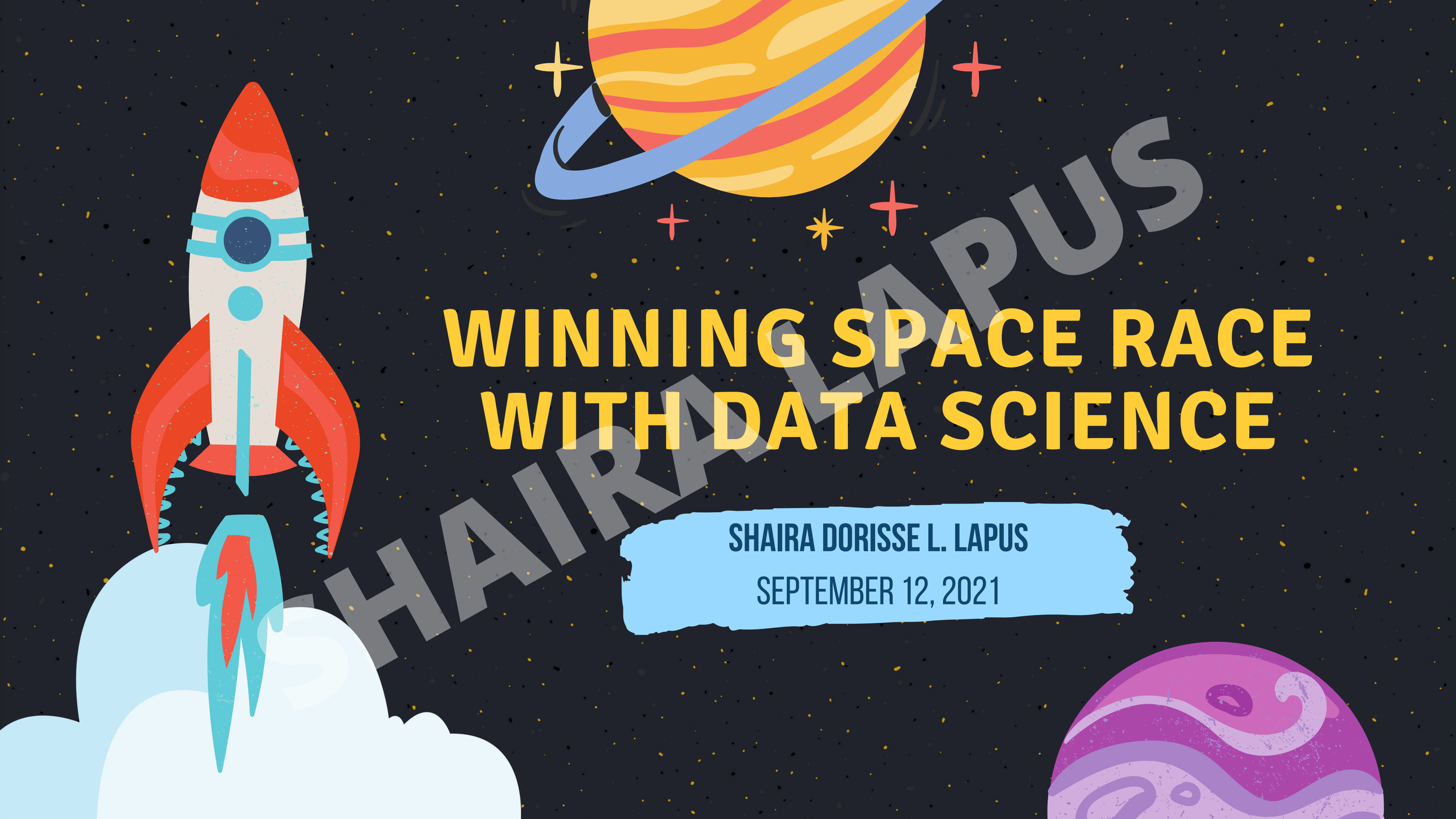


WINNING SPACE RACE WITH DATA SCIENCE

SHAIRA DORISSE L. LAPUS

SEPTEMBER 12, 2021

OUTLINE

EXECUTIVE SUMMARY

Summary of methodologies

- Data Collection via API, SQL, and Web Scraping
- Data Wrangling
- Exploratory Data Analysis (EDA) and Visualization using SQL, Pandas, and Matplotlib
- Interactive Visual Analytics and Dashboard using Folium and Plotly Dash
- Predictive Analysis / Machine Learning Models

Summary of all results

- EDA results and interactive visualization dashboard presented through screenshots
- Comparison of different predictive analysis models

INTRODUCTION

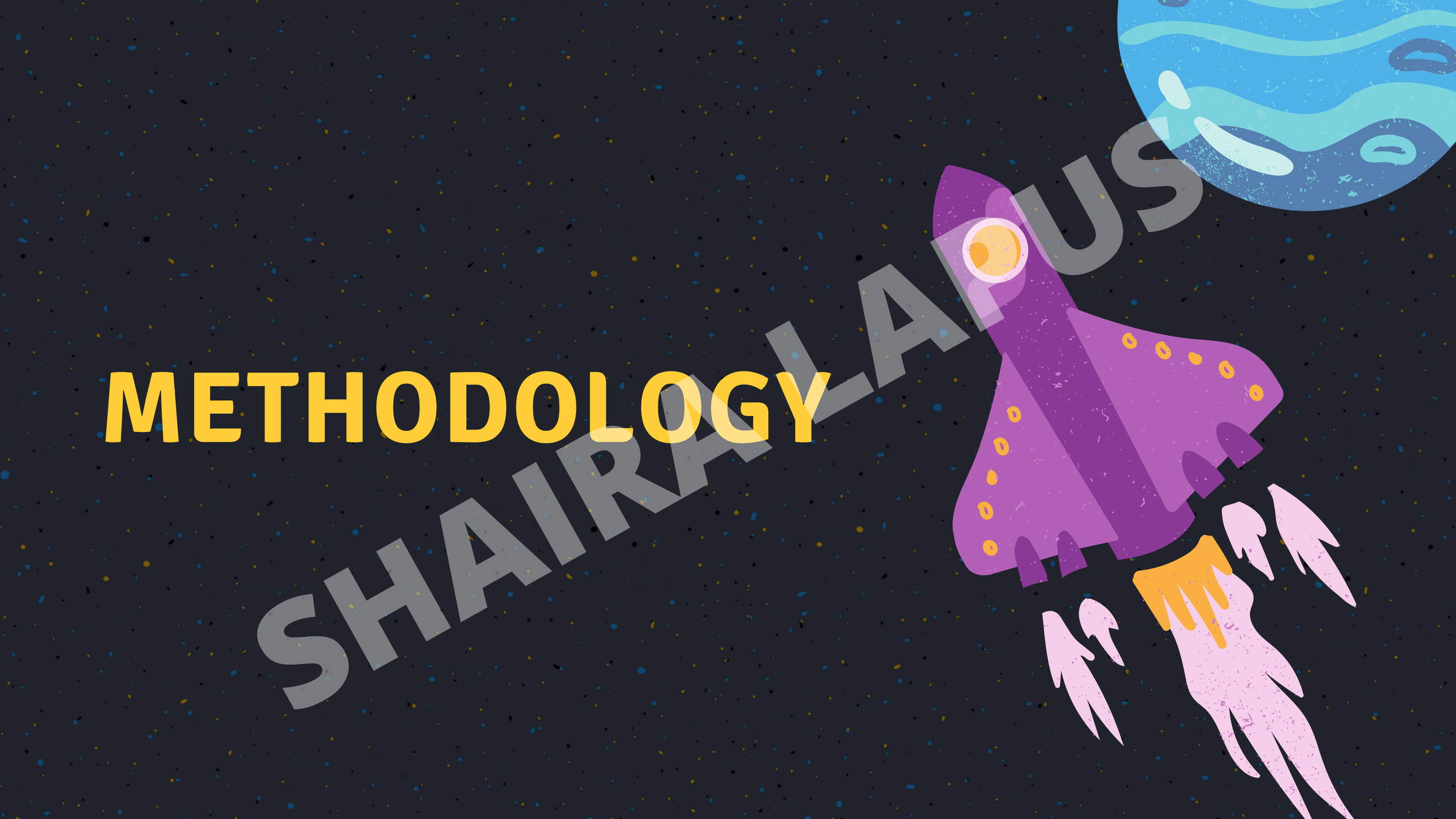
Project Background and Context

This is an analysis of SpaceX's Falcon 9 rocket launches. We want to 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 successfully, this information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Statement of the Problem

- What factors influence a successful landing?
- What conditions will help SpaceX achieve the best rocket landing success rate?
- Which classification model will return the best accuracy in predicting launch outcomes?

METHODOLOGY



METHODOLOGY: EXECUTIVE SUMMARY

Data Collection

The data was collected through the following:

- via SpaceX Rest API
- Web Scraping from Wikipedia

Data Wrangling

- One hot encoding for non-numeric fields
- Dropping irrelevant columns

EDA using visualization and SQL

- SQL queries to summarize different categories
- Scatter plots, bar charts, and line graphs to visualize patterns between variables

Interactive Visual Analytics

- Visual maps using Folium
- Interactive dashboard with pie chart and scatter plot through Plotly Dash which automatically adapts based on the information input by the user

Predictive Analysis using Classification Models

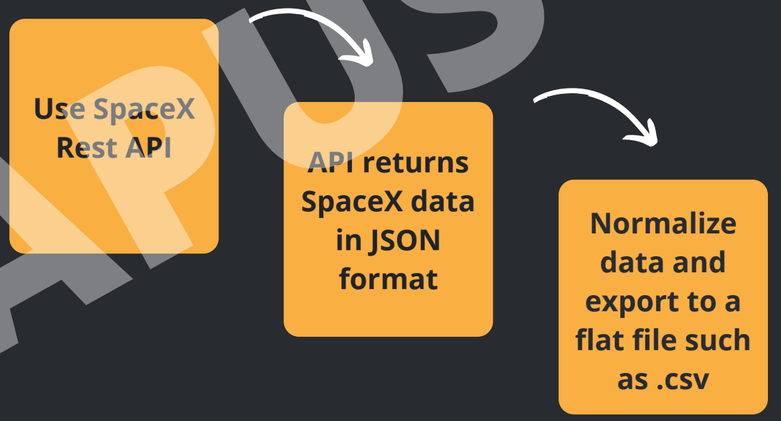
Used the following classification models to predict the outcome of a rocket launch:

- Logistic Regression
- Support Vector Machine (SVM)
- Decision Tree
- K Nearest Neighbors

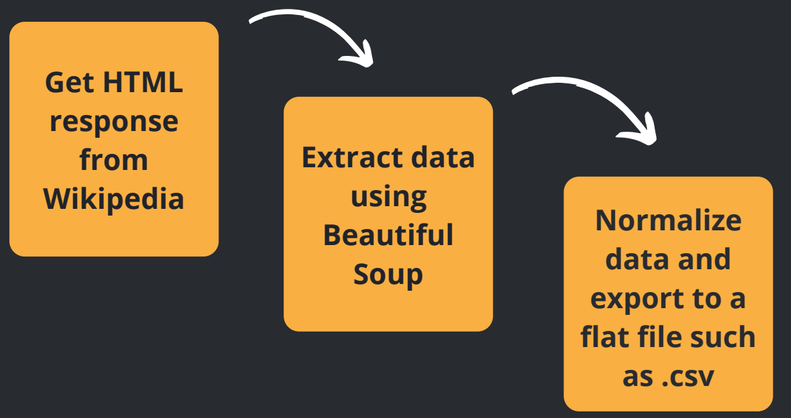
DATA COLLECTION

1. Obtain data from an API or web page
2. Make a dataframe from it
3. Filter dataframe based on requirements
4. Export into a flat file such as .csv
5. Your data is ready for the next steps

METHOD 1: VIA SPACEX API



METHOD 2: VIA WIKIPEDIA WEB PAGE



DATA COLLECTION VIA SPACEX REST API

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

response = requests.get(spacex_url)
```

Getting response from API

1

```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

Applying custom functions to clean data

3

```
df = pd.DataFrame.from_dict(launch_dict)
```

Converting the dictionary into dataframe

5

2

```
response = requests.get(static_json_url).json
data = pd.json_normalize(response.json())
```

Converting response into a .json file

4

Assigning the list to a dictionary

```
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}
```

6

Filtering dataframe and exporting to .csv

```
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
PL_mean = data_falcon9['PayloadMass'].mean()
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, PL_mean)
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GITHUB URL

DATA COLLECTION VIA WEB SCRAPING

Getting response from HTML

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

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

Creating BeautifulSoup object

```
soup = BeautifulSoup(response, "html5lib")
```

Finding tables

```
html_tables = soup.find_all("table")
```

Extracting column names

```
column_names = []

for name in first_launch_table.find_all('th'):
    if name is not None and len(name) > 0:
        column_names.append(extract_column_from_header(name))
```

Dictionary creation and appending data to keys
(see GitHub link for whole code)

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each
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']=[]
```

Converting dictionary to a dataframe

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

Exporting dataframe into .csv

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

GITHUB URL

DATA WRANGLING

Here we remove unnecessary columns and we converted those outcomes into Training Labels: 1 means the booster successfully landed and 0 means it was unsuccessful.

```
df=pd.read_csv("https://cf-courses-data.s3.u
_part_1.csv")
df.head(10)
```

Loading the .csv dataset we made into a dataframe

1

2

Calculating the following:

- No. of launches per site
- No. of occurrence per orbit
- No. and occurrence of mission outcome per orbit type

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

3

Creating a landing outcome label from Outcome column

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

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

```
landing_class = []

for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

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

4

Calculating the success rate

```
df["Class"].mean()
```

5

Exporting into .csv

```
df.to_csv("dataset_part\_2.csv", index=False)
```

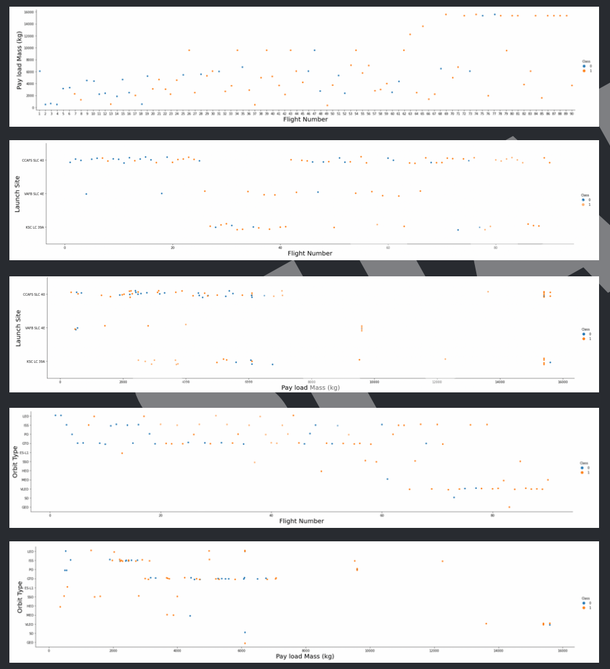

GITHUB URL

EDA WITH DATA VISUALIZATION

SCATTER PLOTS

Scatter plots were used here to show the relationship between two variables, basically how much one is affected by the other. We used scatter plots for the following:

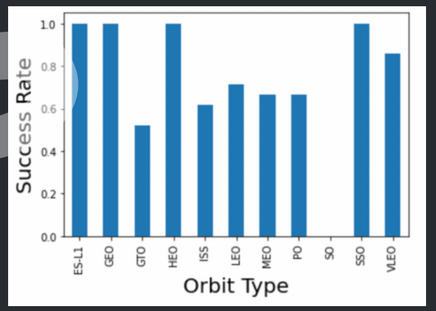
- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type



Zoom in for a clearer view of the graphs

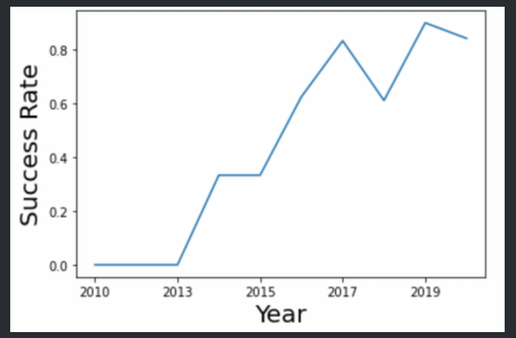
BAR CHART

A bar chart was also used here to show the success rate of each orbit type. Bar charts are helpful in comparing between categories to easily see which performs best.



LINE GRAPH

A line graph is a useful tool in showing trends in a set of data. Here, we used a line graph to plot the success rates per year and we can see that SpaceX's launch success rate has been increasing since 2013.



github url

for the notebook used

EDA WITH SQL

We performed SQL queries to explore our data. Listed below are the tasks and the corresponding SQL queries made.

GITHUB URL

TASKS

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date where the successful landing outcome in drone ship was achieved
- List the names of the boosters which have success in ground pad 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_versionswhich have carried the maximum payload mass
- List the failed landing_outcomesin drone ship, their booster versions, and launch site names for the 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

SQL QUERIES

- %sql SELECT DISTINCT(launch_site) FROM spacextbl
- %sql SELECT * FROM spacextbl WHERE launch_site LIKE 'CCA%' LIMIT 5
- %sql SELECT SUM(payload_mass__kg_) FROM spacextbl WHERE customer = 'NASA (CRS)'
- %sql SELECT AVG(payload_mass__kg_) FROM spacextbl WHERE booster_version = 'F9 v1.1'
- %sql SELECT MIN(date) FROM spacextbl WHERE landing__outcome = 'Success (ground pad)'
- %sql SELECT payload FROM spacextbl WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 and 6000
- %sql SELECT mission_outcome, COUNT(*) FROM spacextbl GROUP BY mission_outcome
- %sql SELECT booster_version FROM spacextbl WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacextbl)
- %sql SELECT landing__outcome, booster_version, launch_site FROM spacextbl WHERE landing__outcome = 'Failure (drone ship)' AND YEAR(DATE) = 2015
- %sql SELECT landing__outcome, COUNT(landing__outcome) FROM spacextbl WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY COUNT(landing__outcome) DESC

BUILD AN INTERACTIVE MAP WITH FOLIUM

GITHUB URL

We used Folium in this project to find geographical patterns among launch sites.
Here we used the following Folium map objects:

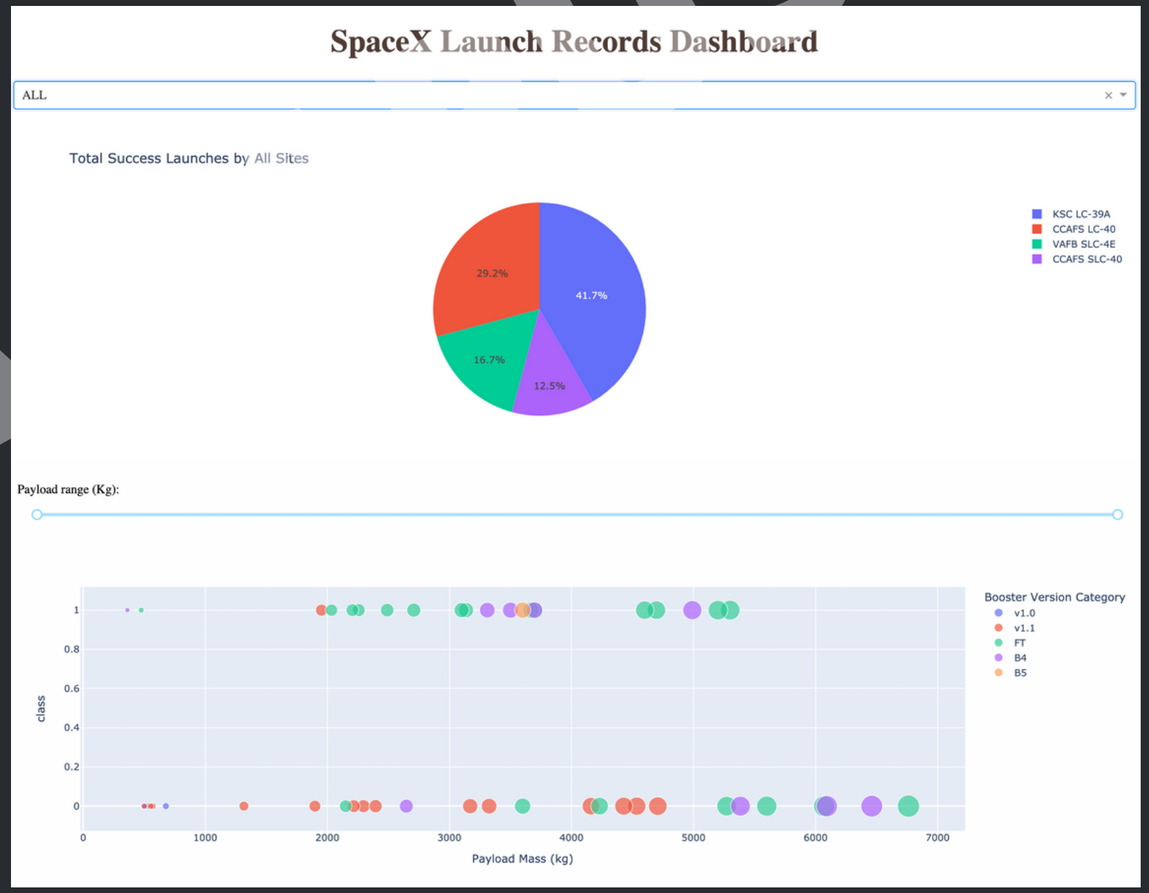
Map Object	Purpose
Circle Marker	Creates a circle to mark the location of interest
Map Marker	Shows a mark on the map for the location we want to show
Icon Marker	Creates an icon on the map to mark the location
PolyLine	Shows a line between two points in the map
Marker Cluster Object	Used to simplify and combine together multiple markers on the map when zoomed out

BUILD A DASHBOARD WITH PLOTLY DASH

GITHUB URL

We used Plotly Dash to create an interactive dashboard containing the following:

- **Dropdown**
  - Used to choose different launch sites
- **RangeSlider**
  - Used to choose a payload range (values between 0kg to 1000kg)
- **Pie Chart**
  - When 'ALL' sites is chosen, it shows the distribution of success launches per site
  - When a specific site is chosen, it shows the distribution of successful vs. failed launches for that site
- **Scatter Plot**
  - We can choosse a payload range using the slider and the booster version categories depending on payload mass and class will show



PREDICTIVE ANALYSIS (CLASSIFICATION)

GITHUB URL

1 **BUILDING THE MODEL**

- Loaded the data into a dataframe
- Transformed it into a NumPy array
- Standardized the data
- Split data into training and testing sets
- Set parameters and created a GridSearchCV object
- Fit object to find the best parameters and train the model

2 **EVALUATING THE MODEL**

- Calculated the accuracy on the test data
- Plotted confusion matrices

3 **IMPROVING THE MODEL**

- Search for the best hyper-parameters for each model using GridSearchCV
- Tested multiple models

4 **FINDING THE BEST MODEL**

- Obtained accuracies for each model's testing sets for comparison

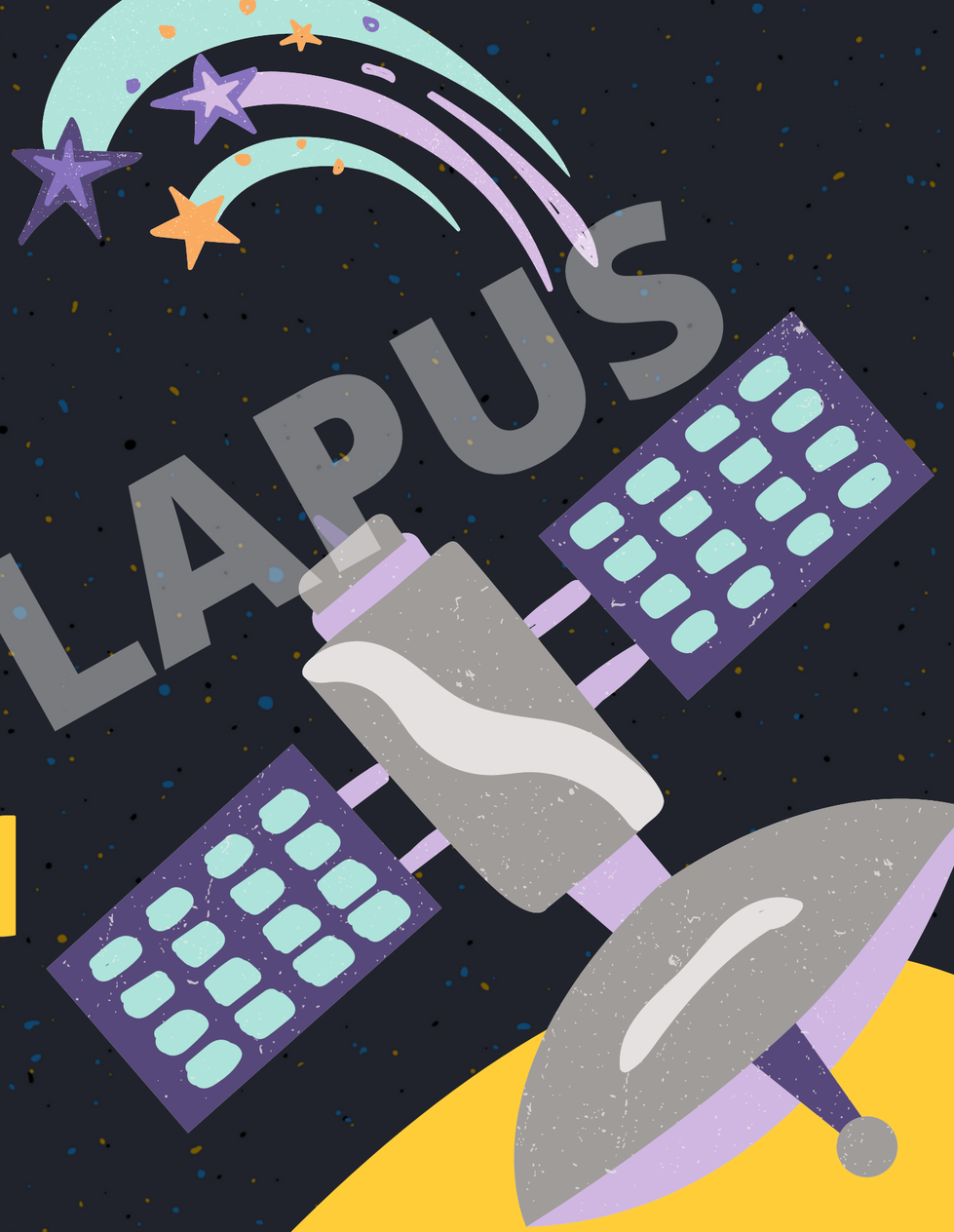
RESULTS

- EXPLORATORY DATA ANALYSIS (EDA) RESULTS
  - EDA WITH VISUALIZATION
  - EDA WITH SQL
- INTERACTIVE VISUAL ANALYTICS DEMO IN SCREENSHOTS
  - FOLIUM
  - DASHBOARD USING PLOTLY DASH
- PREDICTIVE ANALYSIS RESULTS

RESULTS:

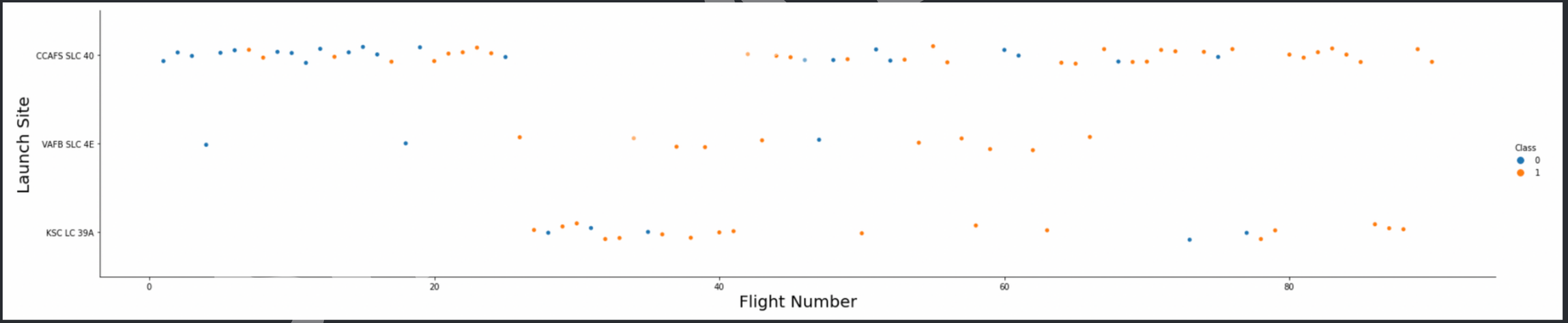
EDA WITH VISUALIZATION

SHAIRA LAPUS



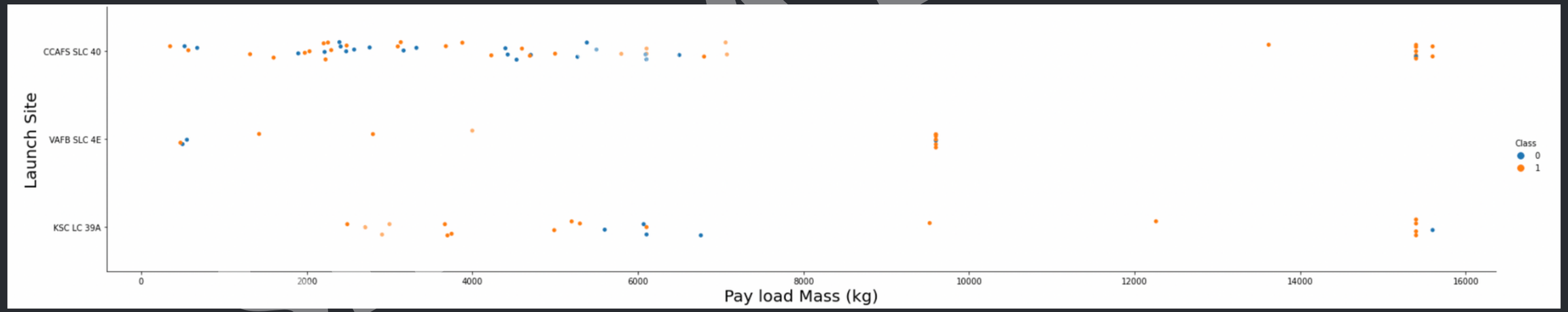
FLIGHT NUMBER VS. LAUNCH SITE

The scatter plot has *Flight Number* in the x-axis and *Launch Site* in the y-axis. The orange dots denote successful landing while the blue dots denote failure. In this scatter plot, we can see that as the flight number increases, the success rate also increases (more orange dots are present). This applies to all three sites mentioned, though CCAFS SLC 40 has more successful landings due to a relatively higher number of launches held at this site compared to the others.



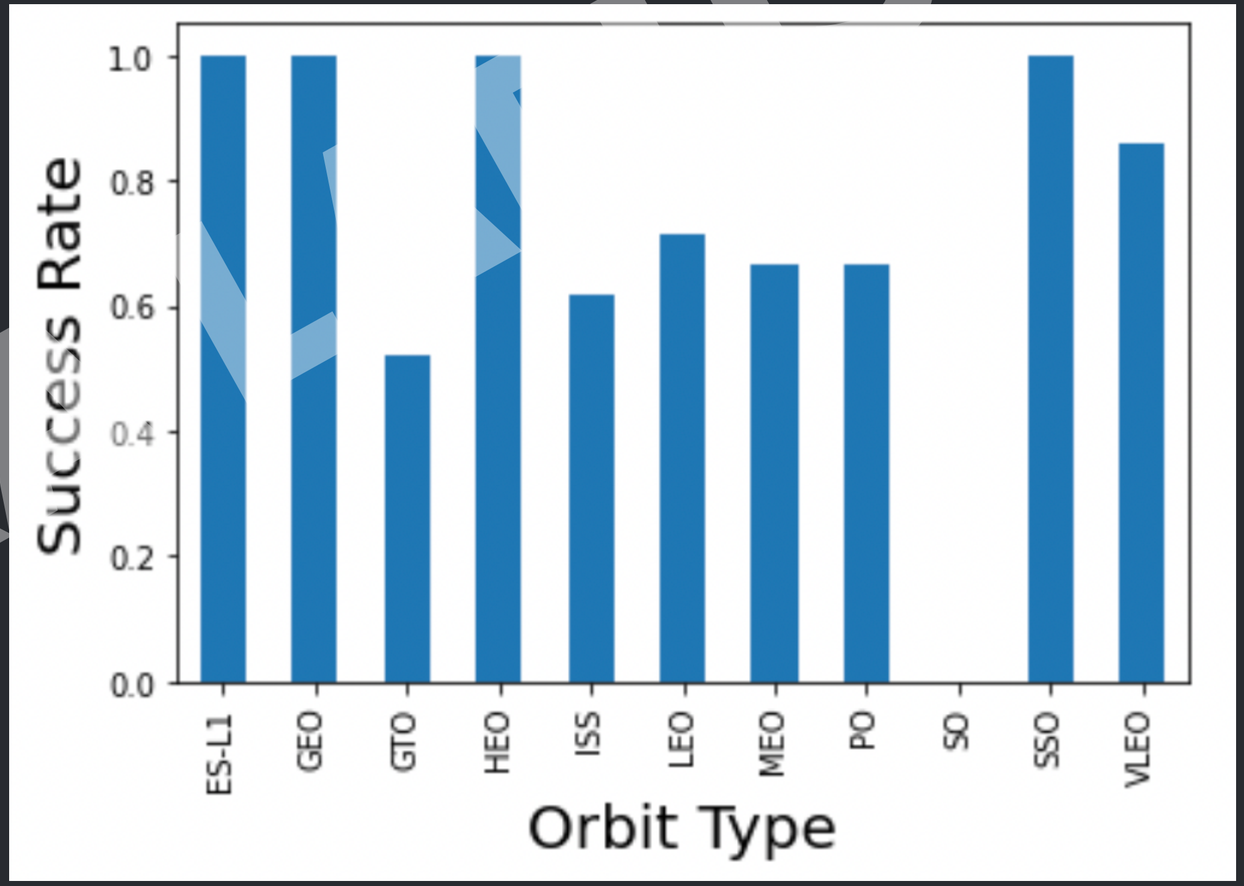
PAYLOAD VS. LAUNCH SITE

The scatter plot has *Payload Mass (kg)* in the x-axis and *Launch Site* in the y-axis. The orange dots denote successful landing while the blue dots denote failure. In this scatter plot, we can see that there are fewer launches being done as the payload mass increases. Success rate also seems to be higher. This trend is more visible in site VAFB SLC 4E, where as the payload mass increases, the success rate also increases. However, there is no obvious trend to assume that the two features are dependent of each other.



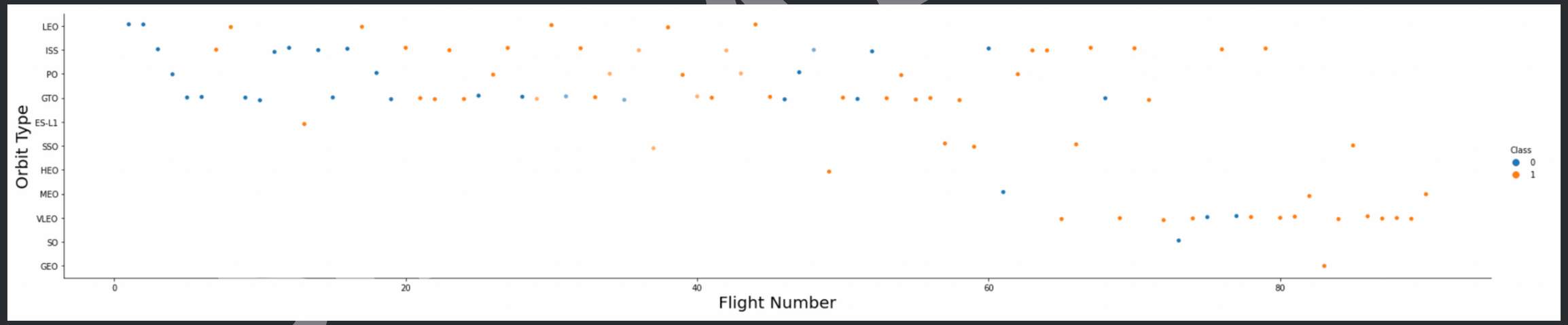
SUCCESS RATE VS. ORBIT TYPE

The bar chart has *Orbit Type* in the x-axis and *Success Rate* in the y-axis. We can see that the orbit types ES-L1, GEO, HEO, and SSO has the highest success rates (100%). Meanwhile, the orbit type SO has 0% success rate.



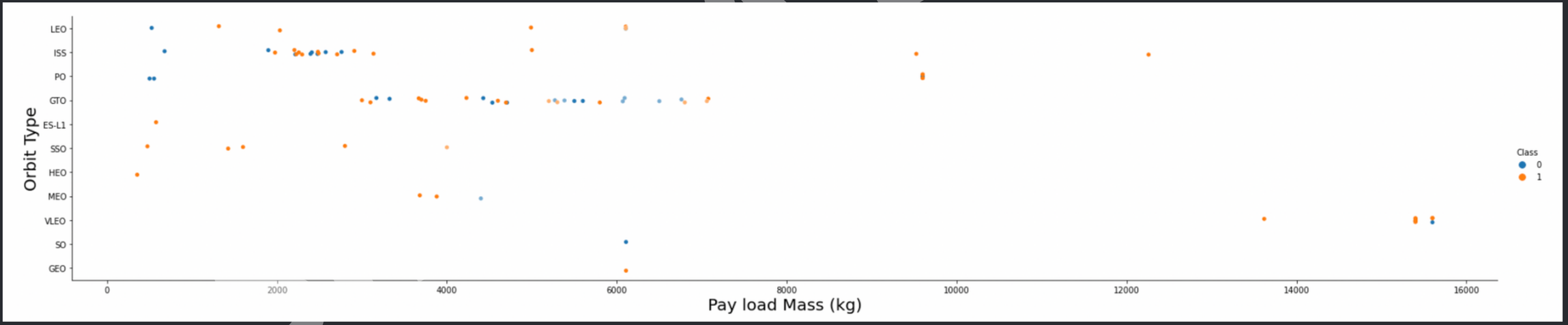
FLIGHT NUMBER VS. ORBIT TYPE

The scatter plot has *Flight Number* in the x-axis and *Orbit Type* in the y-axis. The orange dots denote successful landing while the blue dots denote failure. In this scatter plot, we can see that the success rate for LEO orbit increases as the number of flights increase. On the other hand, there seems to be no trend for GEO orbit as it displays different class values (success vs. failure) as the flight number increases. We can also see that SSO orbit has a 100% success rate in this scatter plot.



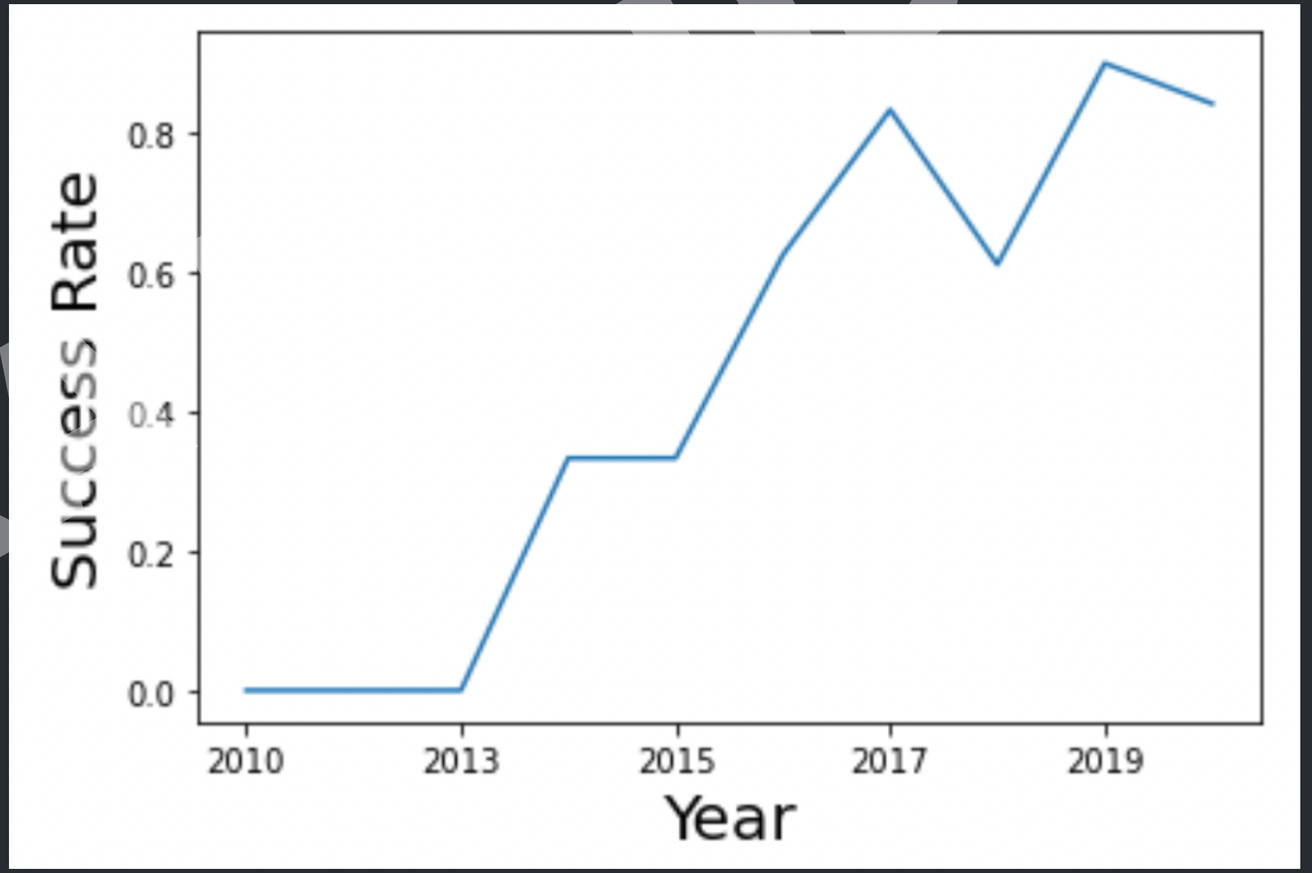
PAYLOAD VS. ORBIT TYPE

The scatter plot has *Payload Mass (kg)* in the x-axis and *Orbit Type* in the y-axis. The orange dots denote successful landing while the blue dots denote failure. In this scatter plot, we can see that the success rate for LEO orbit increases as the payload mass increases. The same happens for ISS orbit. On the other hand, increasing the payload mass affects MEO, GTO, and VLEO orbits negatively–resulting to more failures and therefore, decreasing the success rate for these orbits.



LAUNCH SUCCESS YEARLY TREND

In the line graph, we can see that the success rate of SpaceX's launch landings has been increasing since 2013 to 2019, giving an upward trend to the graph in general. However, it has taken a slight dip in 2020.



RESULTS:

EDA WITH SQL

ALL LAUNCH SITE NAMES

THIS QUERY:

```
%sql SELECT DISTINCT(launch_site) FROM spacextbl
```

WILL RETURN THIS:

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

EXPLANATION:

Using the **DISTINCT()** function will return only the unique elements of the column you chose to return, in this case, the unique launch sites.

LAUNCH SITE NAMES BEGINNING WITH 'CCA'

THIS QUERY:

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

EXPLANATION:

We used the **LIKE** keyword accompanied by **'CCA%'** to search for all launch site names beginning with 'CCA' and specifying the **LIMIT** will return a selected number of rows.

WILL RETURN THIS:

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

TOTAL PAYLOAD MASS

THIS QUERY:

```
%sql SELECT SUM(payload_mass__kg_) FROM spacextbl WHERE customer = 'NASA (CRS)'
```

WILL RETURN THIS:

1
45596

column name, can be changed using 'AS' keyword

EXPLANATION:

Using the **SUM()** function will return the sum of the column chosen and specifying a condition after the WHERE clause will restrict the results.

AVERAGE PAYLOAD MASS BY F9 V1.1

THIS QUERY:

```
%sql SELECT AVG(payload_mass__kg_) FROM spacextbl WHERE booster_version = 'F9 v1.1'
```

WILL RETURN THIS:

1
2928

column name, can be changed using 'AS' keyword

EXPLANATION:

Using the **AVG()** function will return the average of the column chosen and specifying a condition after the WHERE clause will restrict the results.

FIRST SUCCESSFUL GROUND LANDING DATE

THIS QUERY:

```
%sql SELECT MIN(date) FROM spacextbl WHERE landing__outcome = 'Success (ground pad)'
```

WILL RETURN THIS:

1
2015-12-22

column name, can be changed using 'AS' keyword

EXPLANATION:

Using the **MIN()** function will return the minimum value of the column chosen and specifying a condition after the WHERE clause will restrict the results.

SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000

THIS QUERY:

```
%sql SELECT payload FROM spacextbl WHERE landing__outcome = 'Success (drone ship)' AND payload_mass__kg_ BETWEEN 4000 and 6000
```

WILL RETURN THIS:

payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

EXPLANATION:

Using the **AND** keyword allows you to specify more than one argument or condition, and the **BETWEEN-AND** keywords help you filter values between a specified range.

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES

THIS QUERY:

```
%sql SELECT mission_outcome, COUNT(*) FROM spacextbl GROUP BY mission_outcome
```

WILL RETURN THIS:

mission_outcome	2
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

EXPLANATION:

The **COUNT()** function returns the total count of the selected column and the **GROUP BY** keyword groups the same categories together. Here we can see that there are 2 success results, which we can further group together to obtain 100 sucess; 1 failure.

BOOSTERS CARRIED MAXIMUM PAYLOAD

THIS QUERY:

```
%sql SELECT booster_version FROM spacextbl WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM spacextbl)
```

WILL RETURN THIS:

booster_version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

EXPLANATION:

Using the **MAX()** function returns the maximum value of a column. We need to use a subquery here since we cannot call the **MAX()** function directly after the **WHERE** clause.

2015 LAUNCH RECORDS

THIS QUERY:

```
%sql SELECT landing__outcome, booster_version, launch_site FROM spacextbl WHERE landing__outcome = 'Failure (drone ship)' AND
YEAR(DATE) = 2015
```

WILL RETURN THIS:

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

EXPLANATION:

Here we selected three columns to be shown and specified two conditions after the **WHERE** clause using the **AND** keyword. We can see that only two rows/results satisfy the query.

RANK LANDING OUTCOMES BETWEEN 2010-06-04 AND 2017-03-20

THIS QUERY:

```
%sql SELECT landing__outcome, COUNT(landing__outcome) FROM spacextbl WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER BY COUNT(landing__outcome) DESC
```

WILL RETURN THIS:

landing__outcome	2
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

EXPLANATION:

Using SQL queries, we were able to summarize the landing outcomes between two dates. We ranked the results by using the ORDER BY keyword along with the DESC keyword. This returned a list of unique landing outcomes with its number of occurrences.

RESULTS:

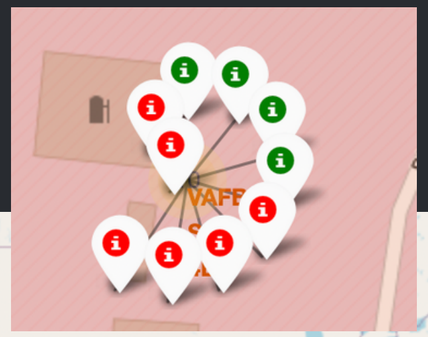
LAUNCH SITES PROXIMITIES ANALYSIS

ALL SPACEX LAUNCH SITES

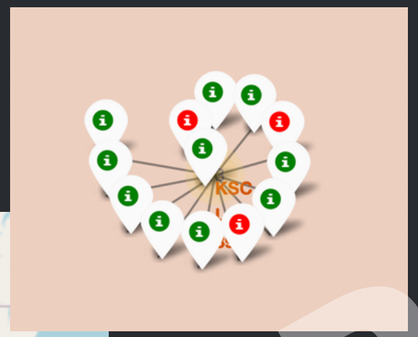
The launch sites' locations are concentrated in the USA. It can also be seen that these launch sites are near coast lines.

ZOOMING IN: LAUNCH OUTCOMES PER SITE

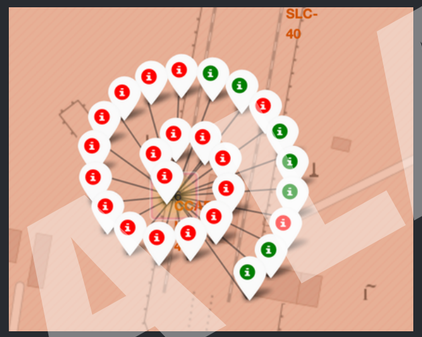
VAFB SLC-4E



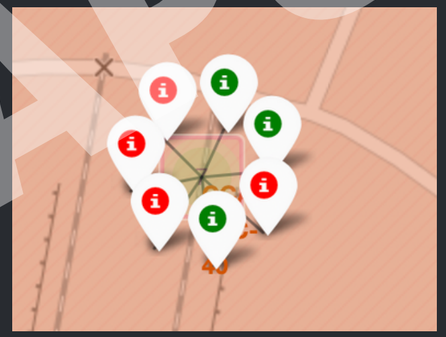
KSC LC-39A

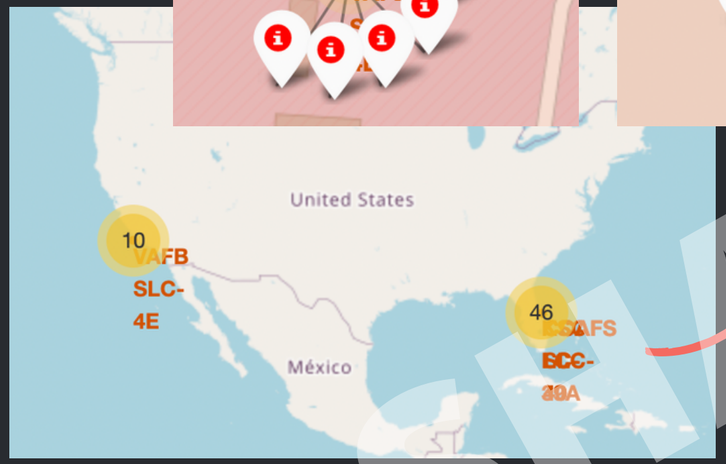


CCAFS LC-40



CCAFS SLC-40

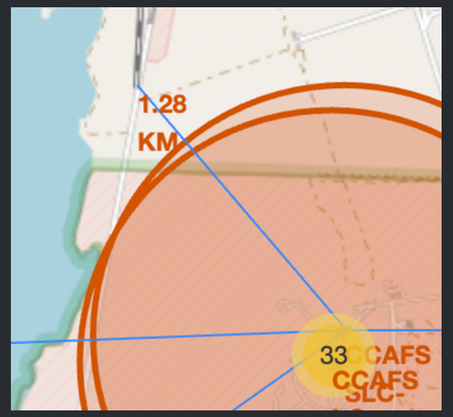




At a closer look, through the color-labeled outcome markers, we can easily see which sites have a higher success rate than others. Here we can see that KSC LC-39A showed the highest success rate when compared to others and oppositely, CCAFS LC-40 showed the highest number of failures.

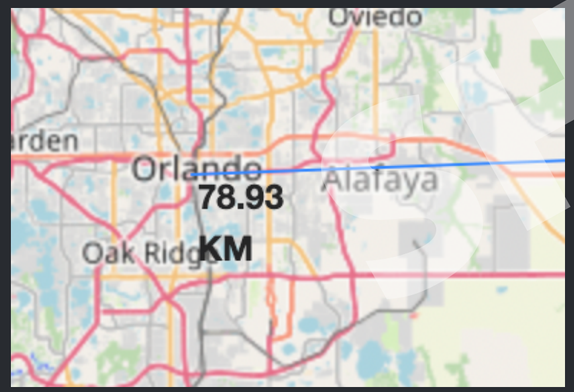
ZOOMING IN: LAUNCH OUTCOMES

Distance to railway

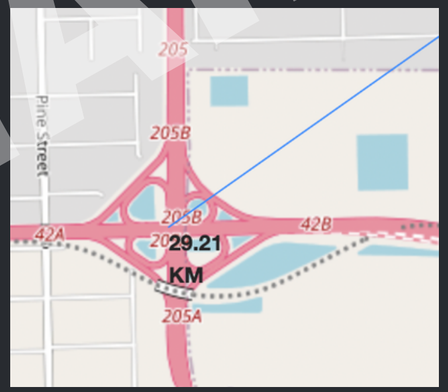


Distance to coast

Distance to nearest city



Distance to nearest highway



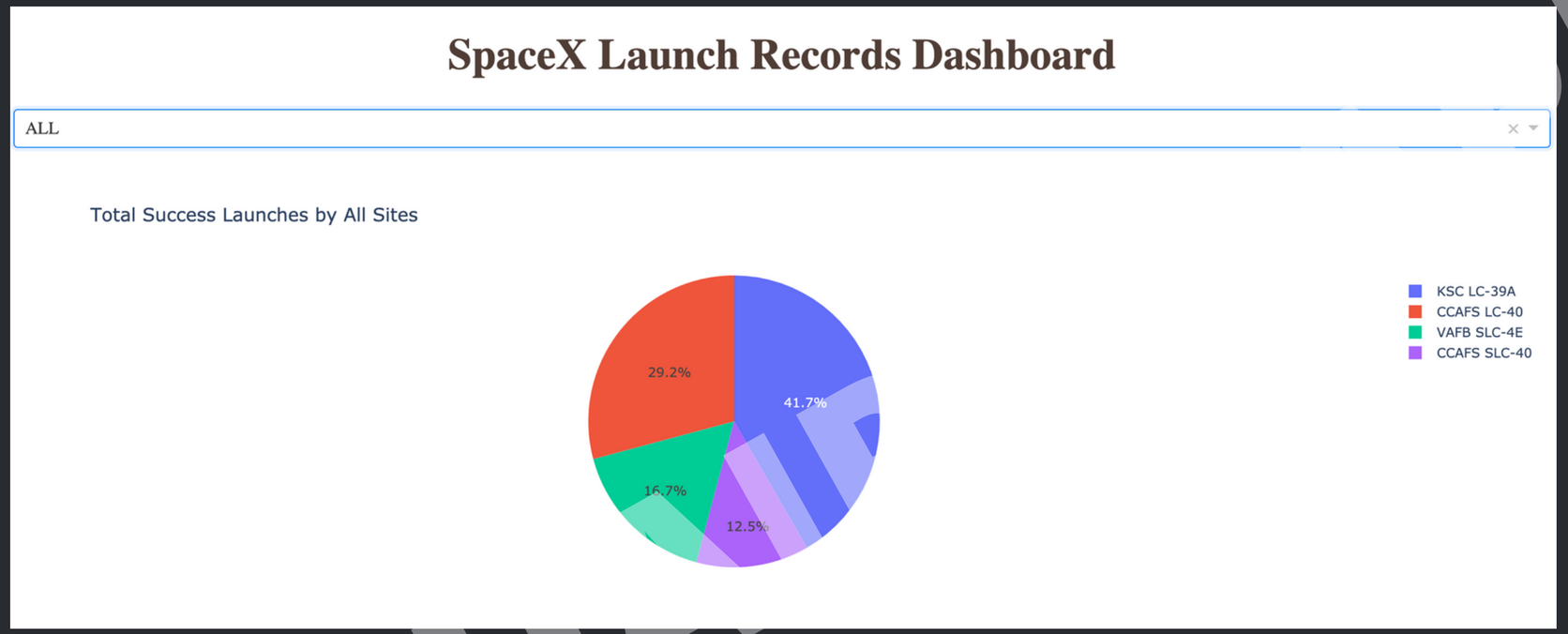
Visualizing the proximity among other locations, we can derive the following:

- Launch sites are located far from cities and highways (within 5 to 80 km)
- Launch sites are generally near railways and coastlines (within 0.5 to 5 km)

RESULTS:

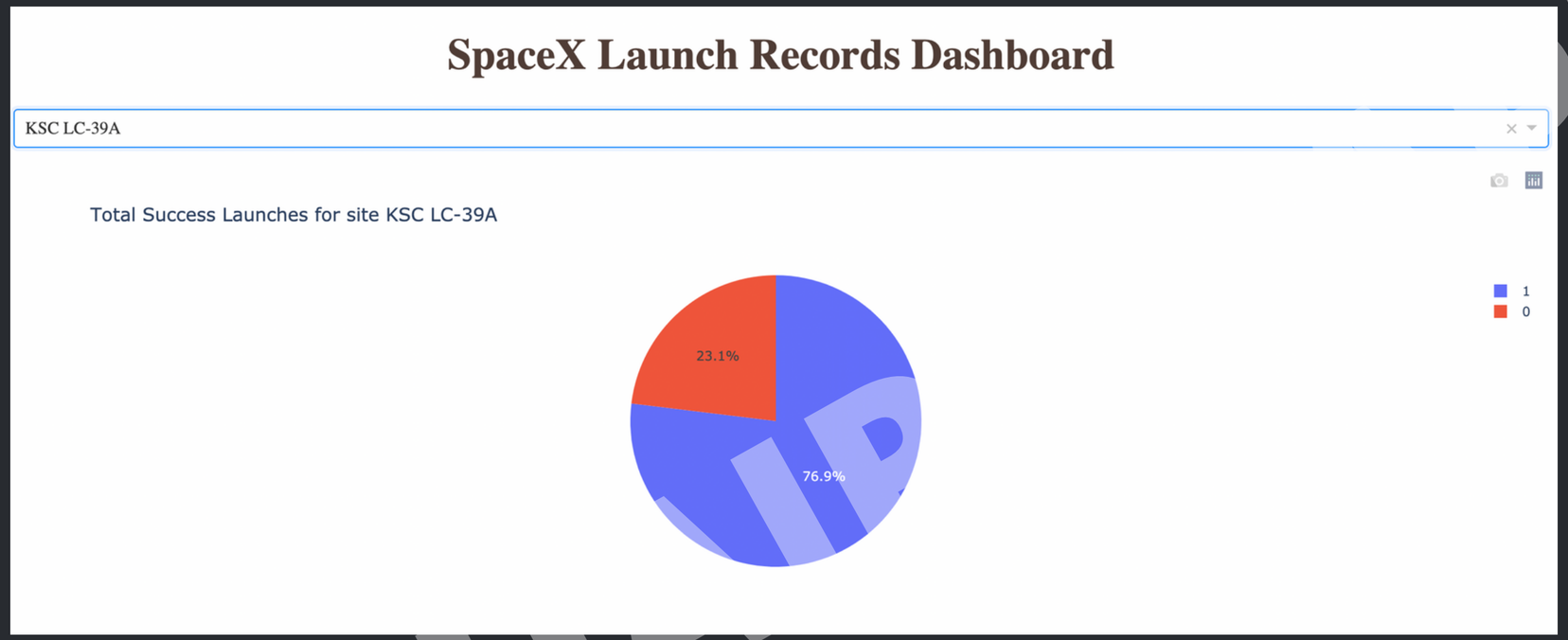
DASHBOARD WITH PLOTLY DASH

LAUNCH SUCCESS COUNT FOR ALL SITES



In this pie chart, we can see that KSC LC-39A has the most number of successful launches, while CCAFS SLC-40 has the least.

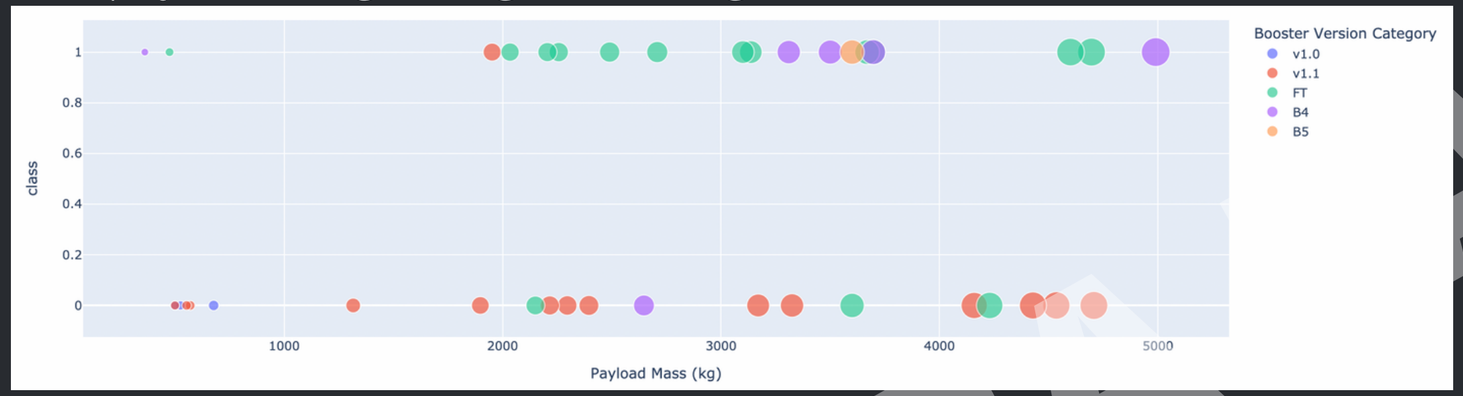
KSC LC-39A SUCCESS RATIO



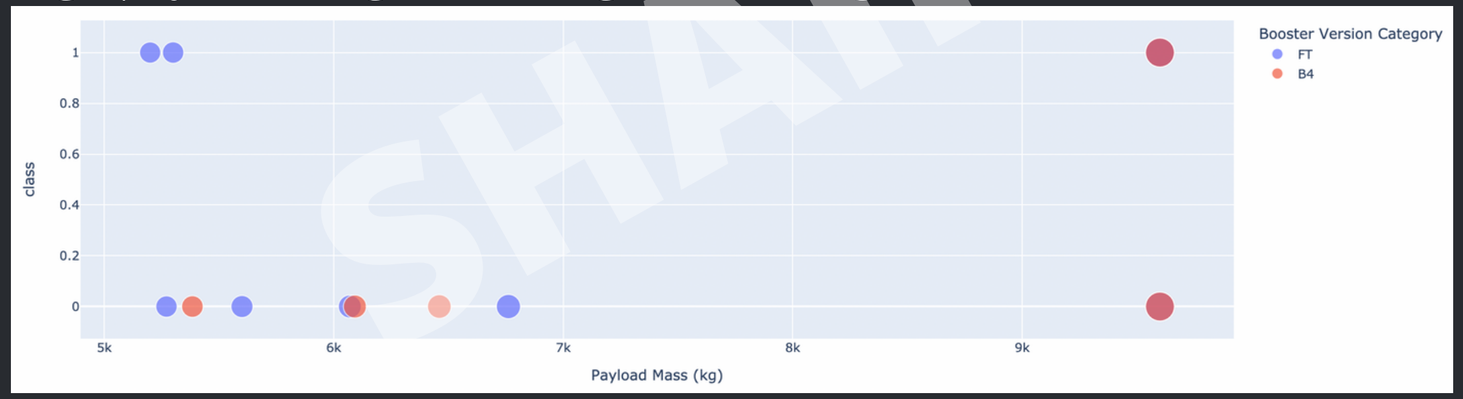
Majority or 76.9% of launches from KSC LC-39A were considered successful and only 23.1% were considered failures.

PAYLOAD VS. LAUNCH OUTCOMES

Low payload weight (0kg to 5,000kg)



High payload weight (5,000kg to 10,000kg)

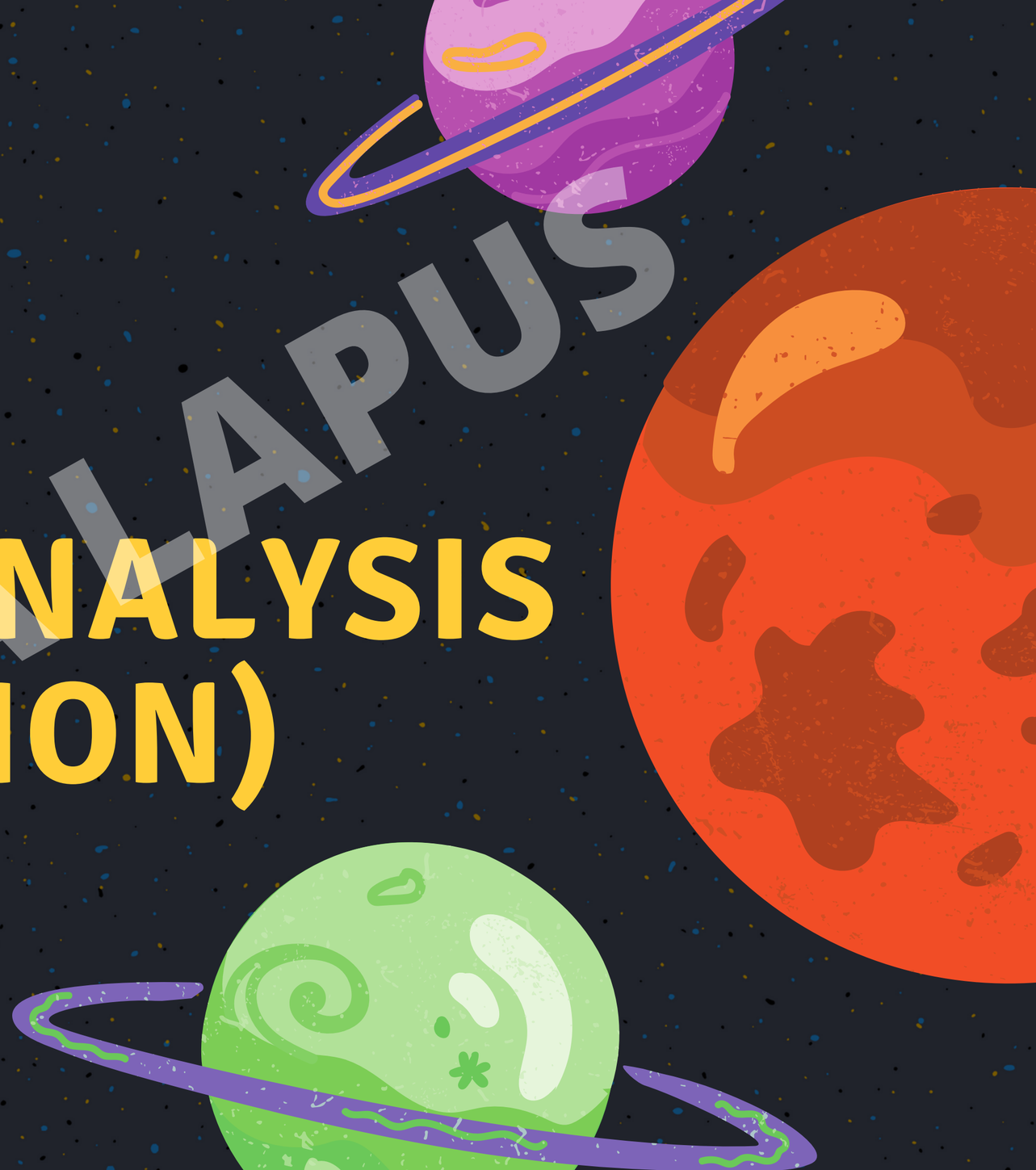


Based on the scatter plots, we can observe the following:

- More launches with low payload weight were conducted
- Success rate is higher for launches with low payload weight
- Only two booster versions were used for launches with heavy payload weights: FT and B4

SECTION 5

PREDICTIVE ANALYSIS (CLASSIFICATION)



LOGISTIC REGRESSION

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

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

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\_`.

```
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
```

Logistic Regression showed an accuracy of 0.85 and the best parameters are shown.

SUPPORT VECTOR MACHINE

```
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 = svm_cv.fit(X_train, Y_train)
```

```
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
```


Support Vector Machine (SVM) showed an accuracy of 0.85 and the best parameters are shown.

DECISION TREE

```
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()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv = tree_cv.fit(X_train, Y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_lea
f': 4, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8767857142857143
```


Decision Tree showed an accuracy of 0.88 and the best parameters are shown.

K NEAREST NEIGHBORS

```
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()
```

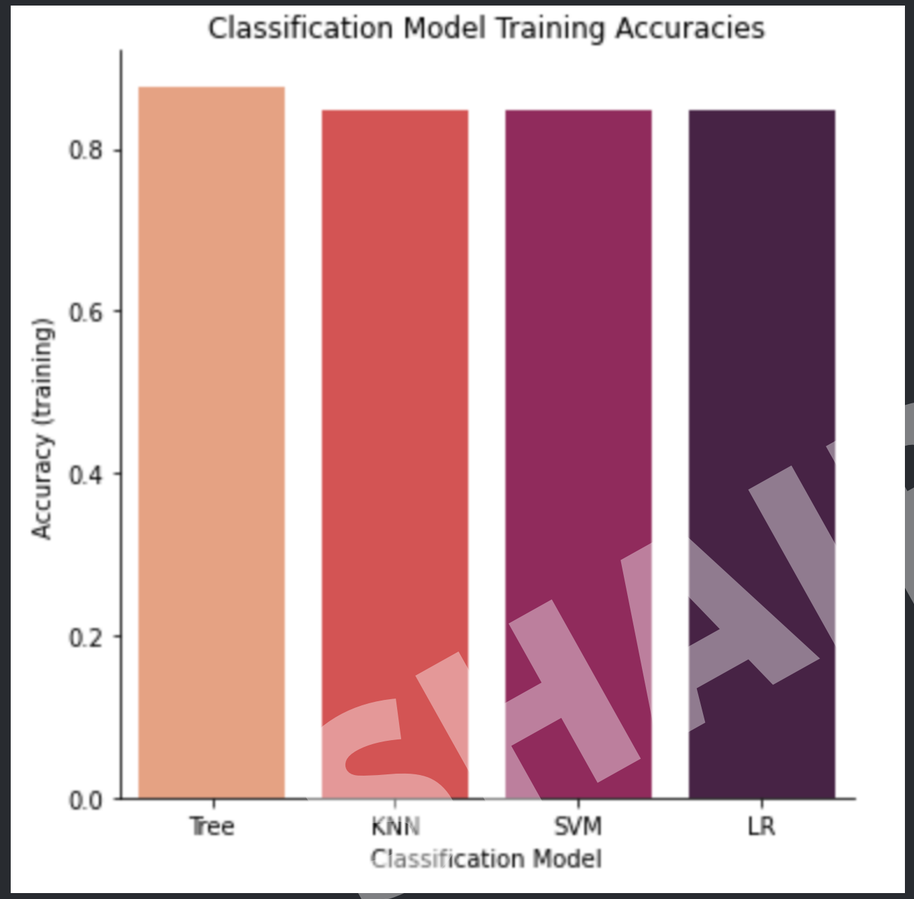
```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv = knn_cv.fit(X_train, Y_train)
```

```
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
```

K Nearest Neighbors showed an accuracy of 0.85 and the best parameters are shown.

CLASSIFICATION ACCURACY



We trained our data using four models: Decision Tree, KNN, SVM, and Logistic Regression. The accuracies of the models on the training set were extremely close however, it is noticeable that the **Decision Tree** model has a slightly higher score among the others (0.88).

When it comes to the testing set, our models all returned an accuracy of 0.83. This is because our dataset is small.

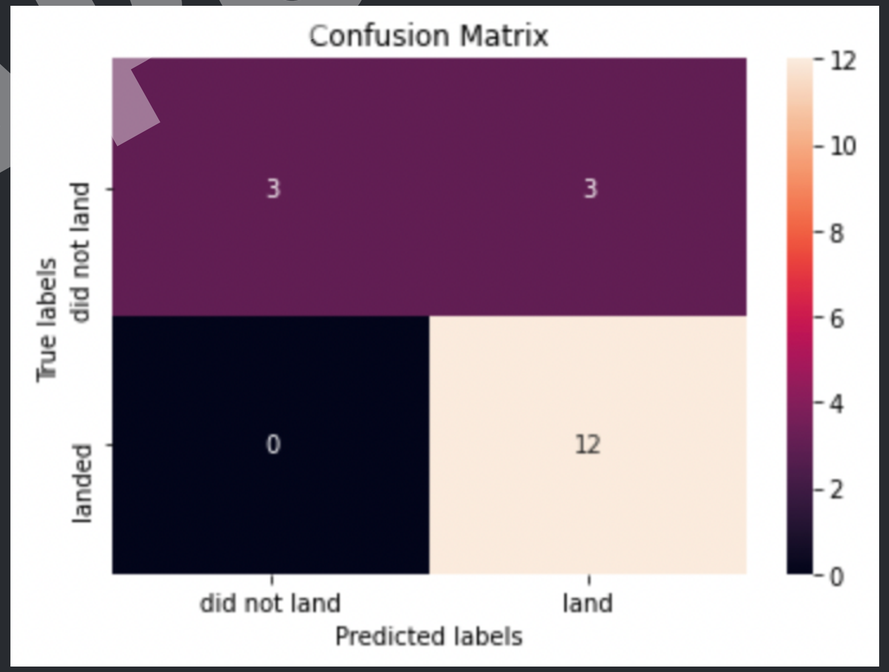
CONFUSION MATRIX

Through the confusion matrix, we can calculate the scores of our model to check how it performs

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Our decision tree model returned the following scores:

- Sensitivity: 0.50
- Specificity: 1.00
- Precision: 1.00
- NPV: 0.80
- Accuracy: 0.83



Decision Tree Confusion Matrix

The scores tell us that the model is good in predicting true negatives (specificity) however, it has a low sensitivity/recall score which shows the model's ability to predict true positives. Even so, the model has high precision and NPV which tells us that the proportion of the data points our model says was relevant actually were relevant.

CONCLUSIONS

1. Factors such as Payload Mass, Orbit Type, and Lauch Site showed effects on the outcome
2. Decreasing the payload mass can positively affect the launches' success rates
3. Launches on orbits ES-L1, GEO, HEO, SSO have the highest success rates
4. Launch site KSC LC-39A has the highest success rate among others and hence further studies on this launch site can be made
5. The decision tree model showed the highest accuracy and is the best model to be used for the dataset

APPENDIX

A. Python code in adding objects to a map via Folium

```
coordinates = [
    [28.56342, -80.57674],
    [28.56342, -80.57674]]

lines=folium.PolyLine(locations=coordinates, weight=1)
site_map.add_child(lines)
distance = calculate_distance(coordinates[0][0], coordinates[0][1], coordinates[1][0], coordinates[1][1])
distance_circle = folium.Marker(
    [28.56342, -80.56794],
    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),
        )
    )
site_map.add_child(distance_circle)
site_map
```

APPENDIX

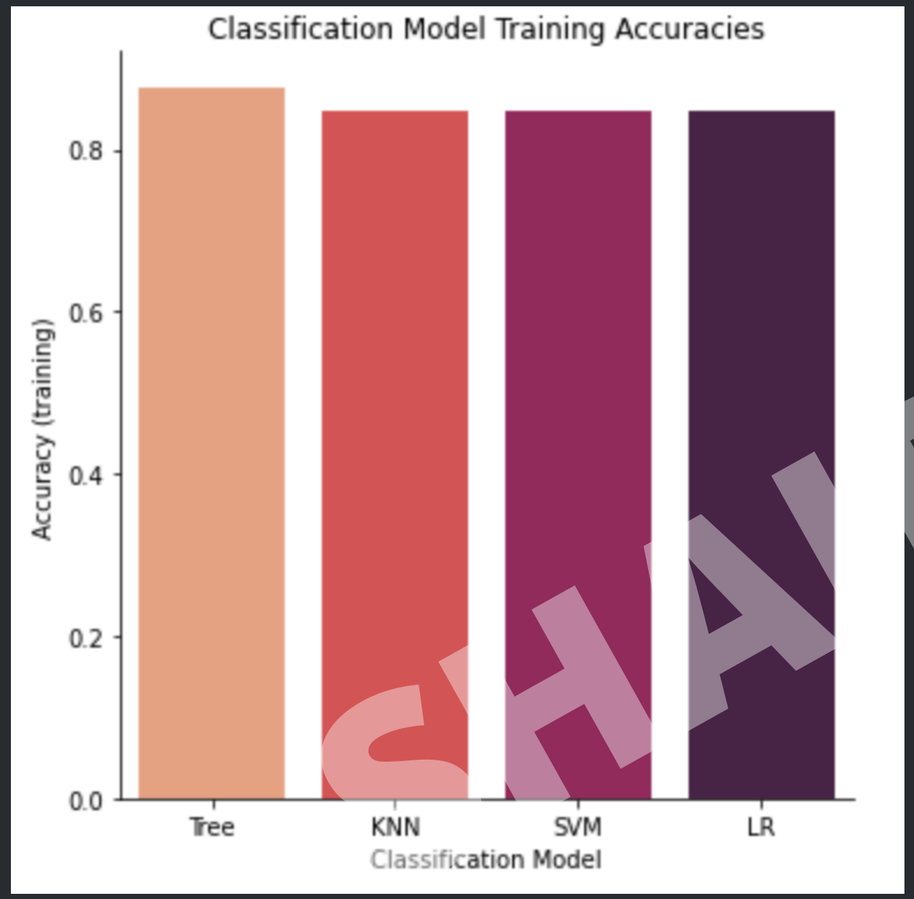
B. Custom functions to create a pie chart and scatter plot for the dashboard

```
def pie_chart(site_dropdown):
    if (site_dropdown == 'ALL'):
        df  = spacex_df[spacex_df['class'] == 1]
        fig = px.pie(df, names = 'Launch Site',title = 'Total Success Launches by All Sites')
    else:
        df  = spacex_df.loc[spacex_df['Launch Site'] == site_dropdown]
        fig = px.pie(df, names = 'class',title = 'Total Success Launches for site ' + site_dropdown)
    return fig
```

```
def scatter_plot(site_dropdown, payload_slider):
    if site_dropdown == 'ALL':
        low, high = payload_slider
        mask = (spacex_df['Payload Mass (kg)'] > low) & (spacex_df['Payload Mass (kg)'] < high)
        fig = px.scatter(
            spacex_df[mask], x="Payload Mass (kg)", y="class",
            color="Booster Version Category",
            size='Payload Mass (kg)')
    else:
        low, high = payload_slider
        df  = spacex_df.loc[spacex_df['Launch Site'] == site_dropdown]
        # df  = spacex_df['Launch Site'] == site_dropdown
        mask = (df['Payload Mass (kg)'] > low) & (df['Payload Mass (kg)'] < high)
        fig = px.scatter(
            df[mask], x="Payload Mass (kg)", y="class",
            color="Booster Version Category",
            size='Payload Mass (kg)')
    return fig
```

APPENDIX

C. Accuracy Bar Chart



Loading the data into a dataframe:

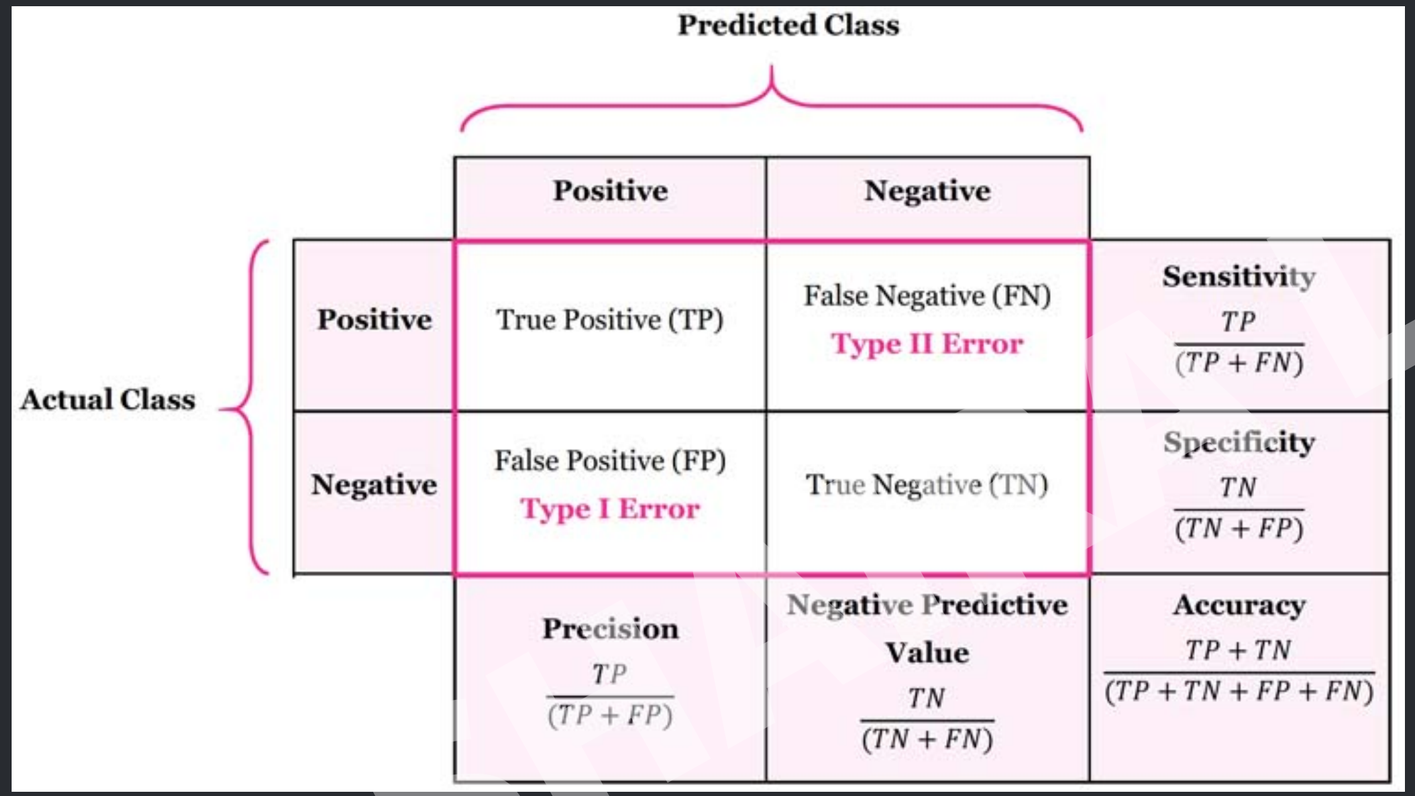
```
models = [['LR', 0.8464285714285713],
['SVM', 0.8482142857142856],
['Tree', 0.8767857142857143],
['KNN', 0.8482142857142858]]

df = pd.DataFrame(models, columns = ['Model', 'Accuracy'])
df
```

Using seaborn to plot the accuracies:

```
ax = sns.catplot(data=df, x='Model', y='Accuracy', kind='bar',
                 order=['Tree', 'KNN', 'SVM', 'LR'], palette='rocket_r')
ax.set(xlabel="Classification Model", ylabel="Accuracy (training)",
       title="Classification Model Training Accuracies")
```

APPENDIX

D. Confusion Matrix Scores



		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Computed the following scores based on the confusion matrix:

- Sensitivity/Recall
- Specificity
- Precision
- Negative Predictive Value
- Accuracy

We can also compute the f1-score which conveys the balance between precision and sensitivity.

Chart obtained from this blog.

THANK YOU!