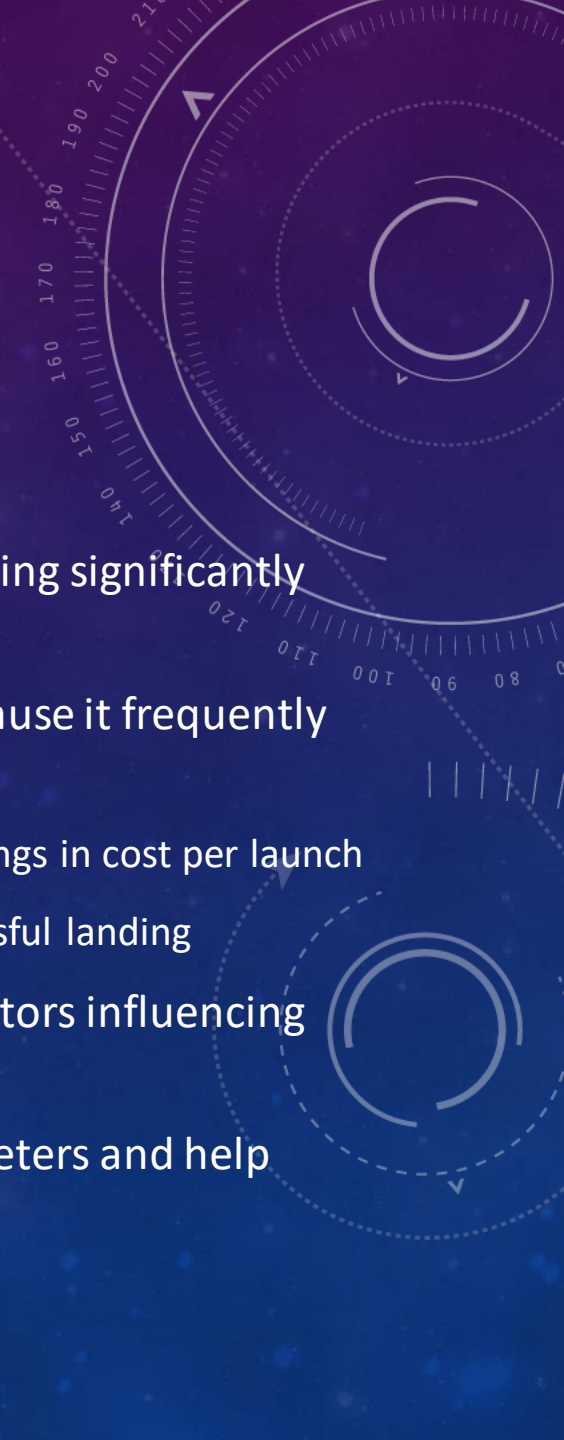


OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
 - Visualization – Charts
 - Dashboard
- Discussion
 - Findings & Implications
- Conclusion
- Appendix

A vertical image of a SpaceX Falcon Heavy rocket launching. The rocket is white with blue "SPACEX" lettering. It is surrounded by a large plume of white smoke and a bright orange and yellow flame at the base. The background is a clear blue sky.

EXECUTIVE SUMMARY

- SpaceX dominates the rocket market by offering significantly lower pricing than competitors
 - SpaceX is able to offer this lower pricing because it frequently reuses the first stage of rockets
 - Reusing the first stage allows substantial savings in cost per launch
 - Reuse of the first stage depends upon successful landing
 - Using publicly available data, we examine factors influencing successful F9 landings.
 - Machine learning models identify key parameters and help predict successful landings.
- 
- In the background of the right side of the slide, there are faint, stylized technical diagrams. These include concentric circles, radial lines, and curved arrows, resembling a radar screen or a control panel interface.

INTRODUCTION

Project Background & Context

- SpaceX advertises Falcon 9 rocket launches at \$62 million
- Competitors cost upwards of \$165 million
- Savings is attributed to reuse of first stage of rocket

Project Task

- Predict factors contributing to successful landing of Space X Falcon 9 rocket





METHODOLOGY

- Data Collection Methodology
 - Obtain publicly available data regarding SpaceX launches from SpaceX REST API and Wikipedia page titled [List of Falcon 9 & Falcon 9 Heavy Launches](#)
- Data Wrangling
 - Prepare the data for analysis by eliminating irrelevant columns, replacing null values, and performing One-Hot encoding.
- Perform exploratory data analysis (EDA) using SQL queries & visualization to observe relationships between variables
- Perform Interactive Visual Analytics using Folium & Plotly Dash
- Perform predictive Analysis using classification models to identify best classifier
 - Logistic Regression, K Nearest Neighbor, SVM, Decision Tree

DATA COLLECTION METHOD

SPACEX REST API

Use Space X
REST API



API returns
JSON data



Normalize data
into csv file

WEB SCRAPING

Get HTML
Response from
Wikipedia



Extract data
using beautiful
soup



Normalize data
into csv file

- SpaceX Launch Data collected from SpaceX Rest API
- The API provides data in CSV format about launches, including rocket booster type, payload delivered, launch specifications, landing specifications, and landing outcome.
- The SpaceX API endpoints (URL) starts with `api.spacexdata.com/v4/`
- We also use BeautifulSoup to perform web scraping on SpaceX's Wikipedia page to obtain launch data

DATA COLLECTION



1. Get a response from API

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

response = requests.get(spacex_url)
```

2. Turn it into Pandas dataframe using .json_normalize()

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

3. Create custom functions for the variables

```
# Call getBoosterVersion
getBoosterVersion(data)
```

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

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

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

4. Create custom functions for the variables

```
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}
```

5. Create a Pandas dataframe from the dictionary launch_dict.

```
# Create a dataframe from launch_dict
launch_data=pd.DataFrame(launch_dict)
```

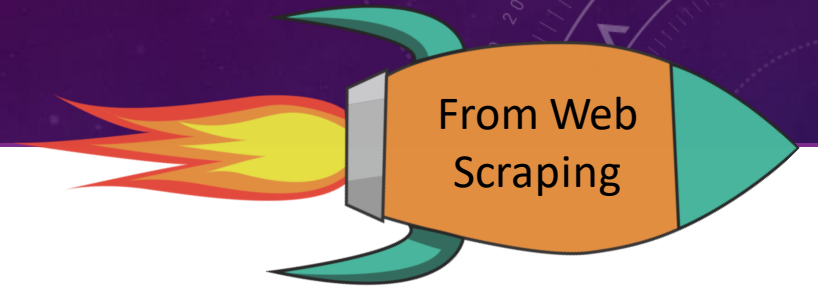
6. Remove Falcon 1 launches from dataframe

```
indexnames = launch_data.loc[launch_data['BoosterVersion']!='Falcon 1'].index
launch_data.drop(indexnames, inplace = True)
data_falcon9 = launch_data
```

7. Replace null values

```
# Calculate the mean value of PayloadMass column
mean = data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.NaN, mean)
data_falcon9
```

DATA COLLECTION



1. Request Falcon9 Launch Wiki page from its URL

```
# use requests.get() method with the provided static_url
data=requests.get(static_url)
```

2. Create BeautifulSoup object from response text

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(data.content, "html.parser")
```

3. Find tables

```
table=soup.find_all('table')
table_row=soup.find_all(name='tr')
```

4. Extract column names

```
column_names = []

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

5. Create dictionary

```
launch_dict=dict.fromkeys(column_names)
# Remove an irrelevant column
del launch_dict['Date and time ( )']
# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Append data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table','wikit
# get table row
for rows in table.find_all("tr"):
    #check to see if first table heading is as number corres
    if rows.th:
        if rows.th.string:
            flight_number=rows.th.string.strip()
            flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictionary
        if flag:
            extracted_row += 1
```

7. Convert dictionary to dataframe

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

8. Save as .csv

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

DATA WRANGLING



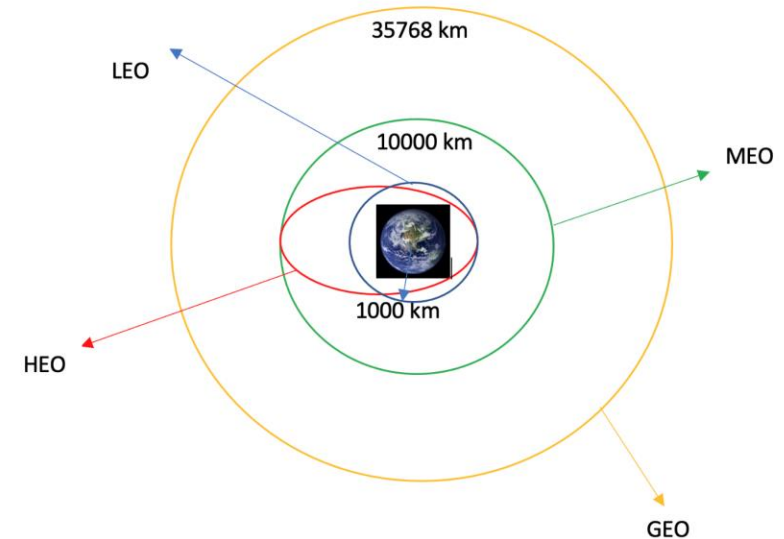
- Clean, organize & transform our raw data
- Determine what we should keep for training supervised models
- Calculate number of launches at each site and the number and occurrence of each orbit.
- Create Landing Outcome label from outcome results to simplify into success or failure.

Launch Site	# Launches
CCAFS SLC 40	55
KSC LC 39A	22
VAFB SLC 4E	13

Success Rate	66%
--------------	-----

Landing Outcome	#	Success
True ASDS	41	Yes
None None	19	No
True RSLs	14	Yes
False ASDS	6	No
True Ocean	5	Yes
False Ocean	2	No
None ASDS	2	No
False RTLS	1	No

Orbit	# Launches
GTO	27
ISS	21
VLEO	14
PO	9
LEO	7
SSO	5
MEO	3
ES-L1	1
HEO	1
SO	1
GEO	1



EXPLORATORY DATA ANALYSIS

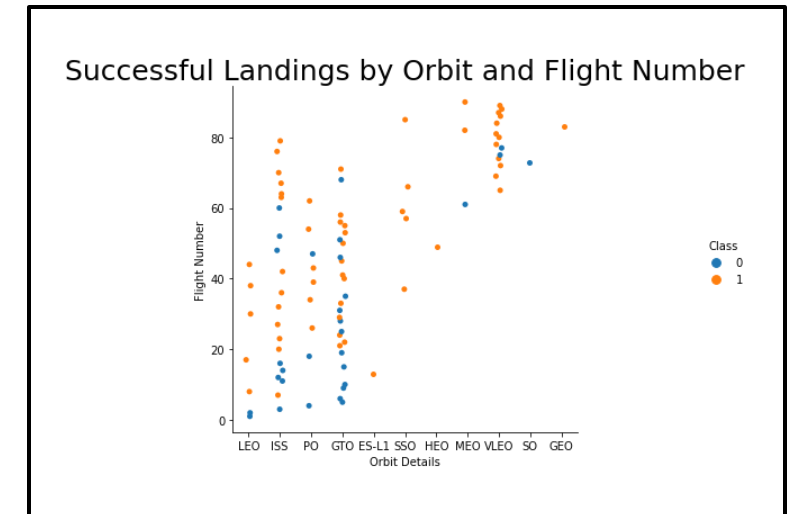
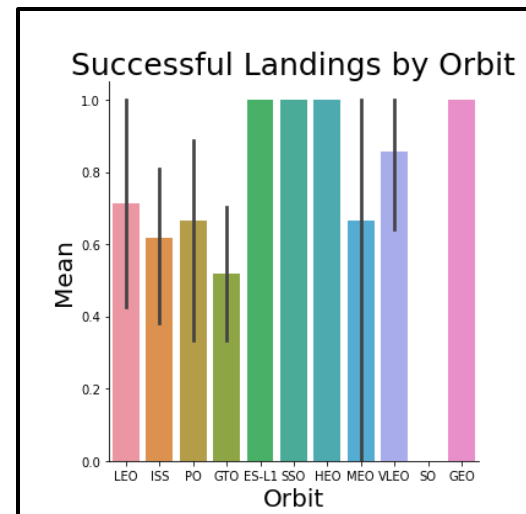
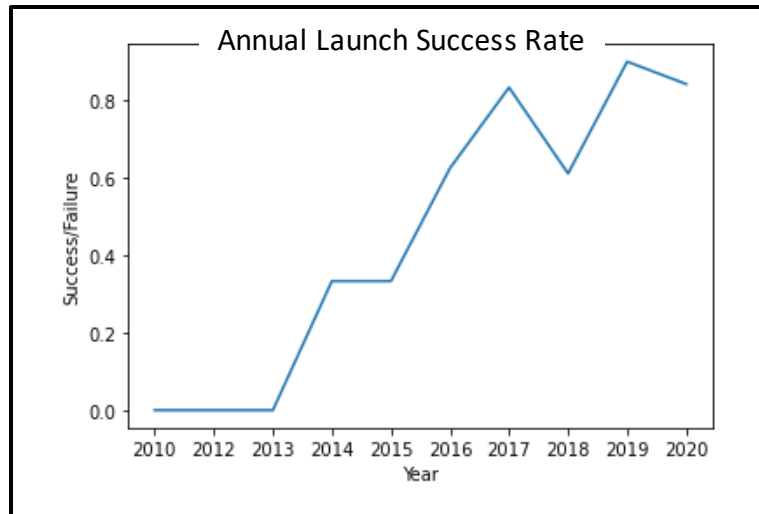
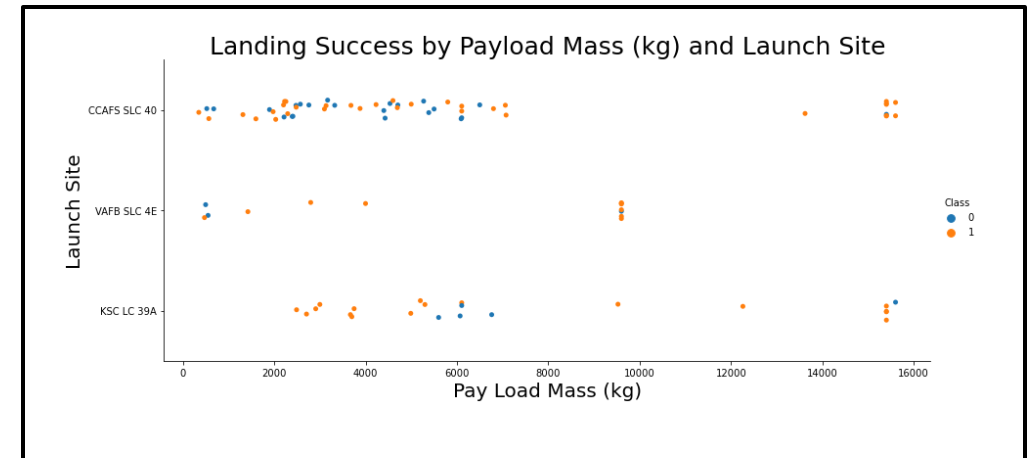


SQL was used to perform the following queries

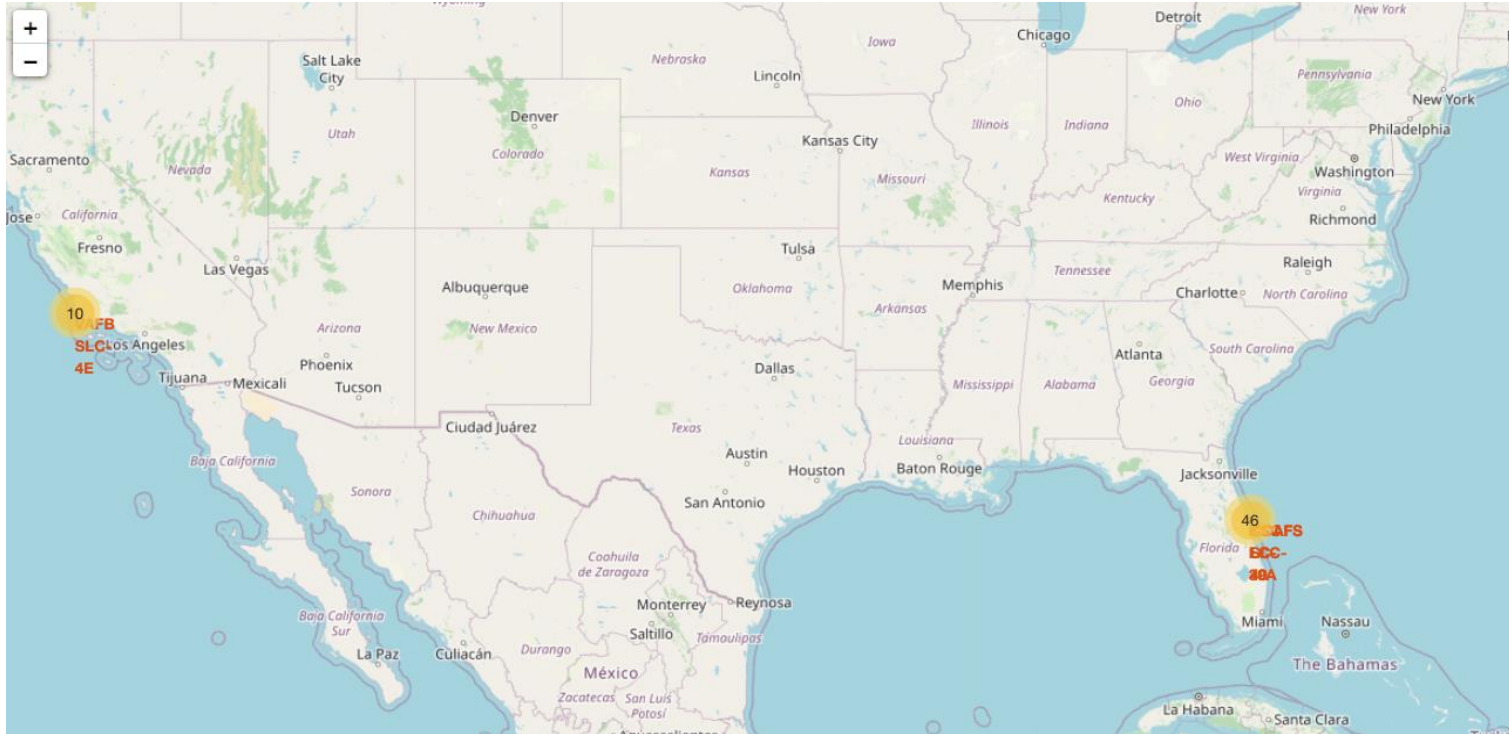
- We utilized SQL magic to perform SQL queries within our jupyter notebook.
- We explored our data to gain insight into patterns hidden within the data, for instance:
 - The names of unique launch sites
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload carried by the booster version F9 v1.1
 - The total number of successful vs. failed missions
 - The failed landing outcomes in drone ship landings, their booster version, and launch site names
- We note that there is a difference between Mission Outcome and Landing Outcome; 99% of missions are successful whereas only 66% of landings are successful.

EDA & INTERACTIVE VISUAL ANALYSIS METHODOLOGY

We explore the data by visualizing the relationship between different flight number and launch site, success rate of each orbit type, flight number and orbit type, and the annual trend in launch success.



FOLIUM INTERACTIVE MAP



- Interactive Maps Built with Folium allows users to drill down into the map examining launch site locations, their surroundings, and success vs. Failures
- We marked all launch sites and added map objects such as markers, circles, and lines to enhance the information we can gather at a glance from the map.
- We used one-hot encoding for the variable "Launch Outcomes" (0=failure, 1= success)
- We employed marker clusters and color-labeled markers to visually indicate launch site success.
- We calculated the distance between a launch site and nearby facilities (railway, highway, city, and coastline)

PREDICTIVE ANALYTICS

1. Data is loaded and read into Pandas dataframes

```
data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.com/resources/cf-courses-data/Data_Dumpers/titanic/train.csv')
data.head()

X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.com/resources/cf-courses-data/Data_Dumpers/titanic/train.csv')
X.head(100)
```

2. Create NumPy array from column 'Class' in data

```
# target data as numpy array
Y = data['Class'].to_numpy()
Y
```

3. Standardize the data

```
# apply standard scaler to features
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

4. Split data X and Y into training/test data

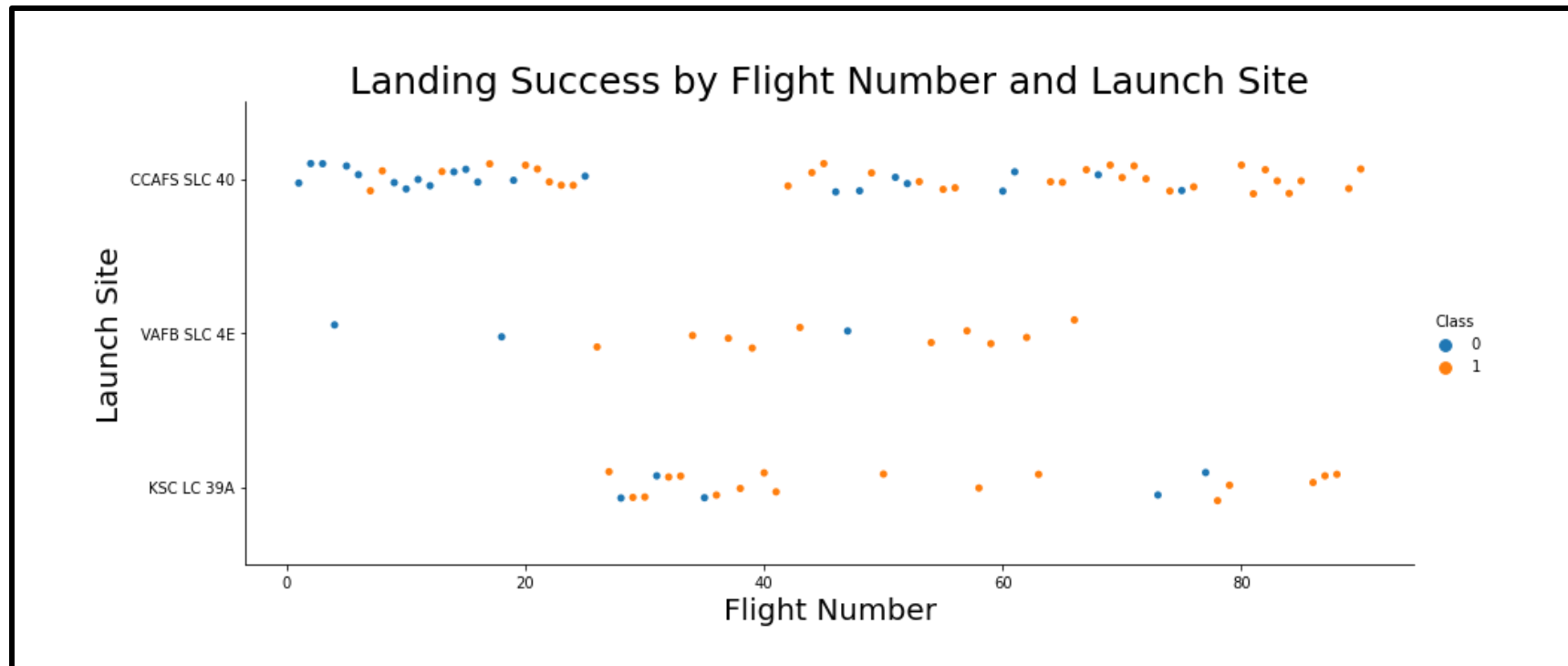
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print('Train set:', X_train.shape, Y_train.shape)
print('Test set:', X_test.shape, Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

- Machine learning is used to train/test various models and find the best hyperparameters using GridSearch CV for SVM, Classification Trees, KNN, and Logistic Regression.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.

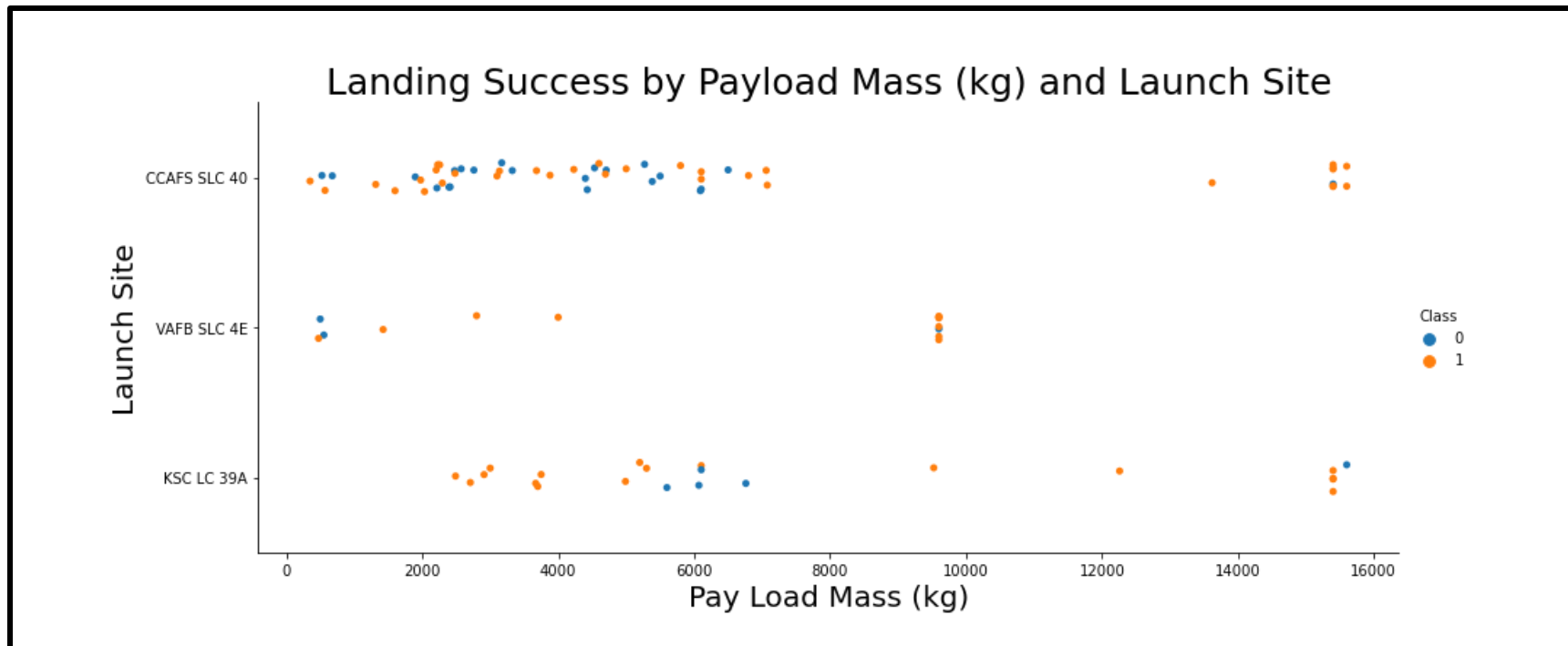
FLIGHT NUMBER BY LAUNCH SITE

- The different launch sites have different success rates
- As the flight number increases, the first stage is more likely to land successfully
- CCAFS SLC 40 has more launches than other sites



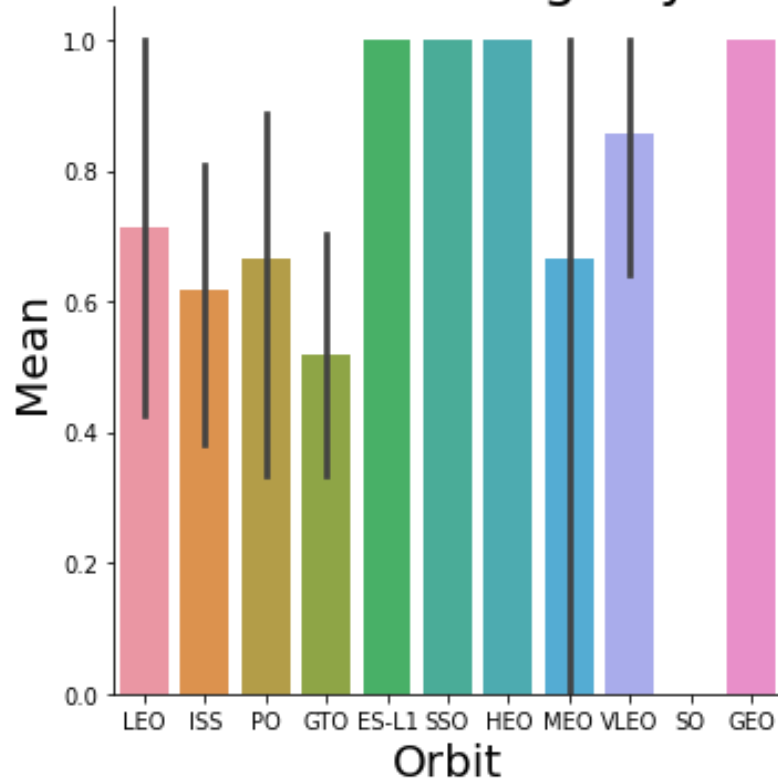
PAYLOAD BY LAUNCH SITE

- There are no rockets launched for heavy payload mass (greater than 10,000 kg) at the VAFB SLC-4E site
- The larger the payload, the less likely the first stage will return.
- The majority of iPay Loads with lower mass have been launched from CCAFS SLC 40.



OUTCOME BY ORBIT

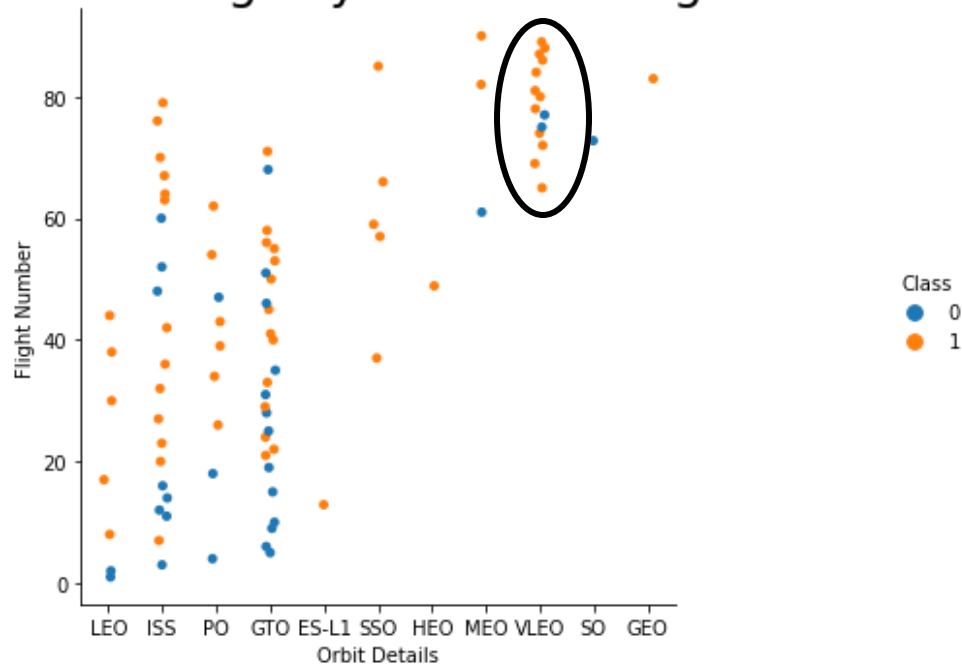
Successful Landings by Orbit



- Several orbits show 100% success rate, but of these, only SSO has more than one launch
- VLEO (85.7%) and LEO (71.4%) have the next highest number of successful landings
- The GTO orbit has a probability of success only slightly better than chance

FLIGHT NUMBER BY ORBIT

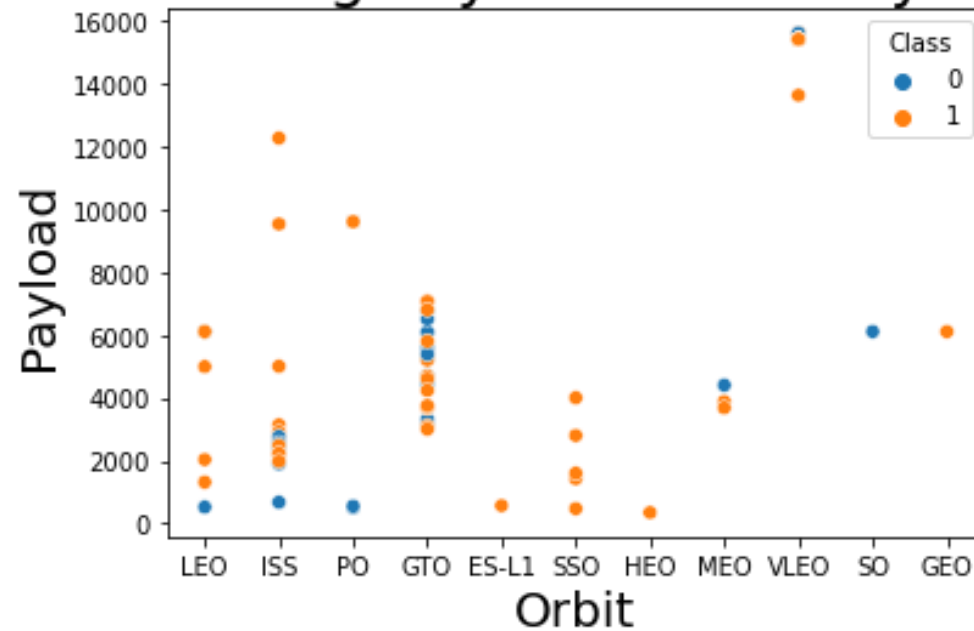
Successful Landings by Orbit and Flight Number



There appears to be a shift in recent years toward more frequent use of the VLEO orbit, perhaps due to the high rate of landing success.

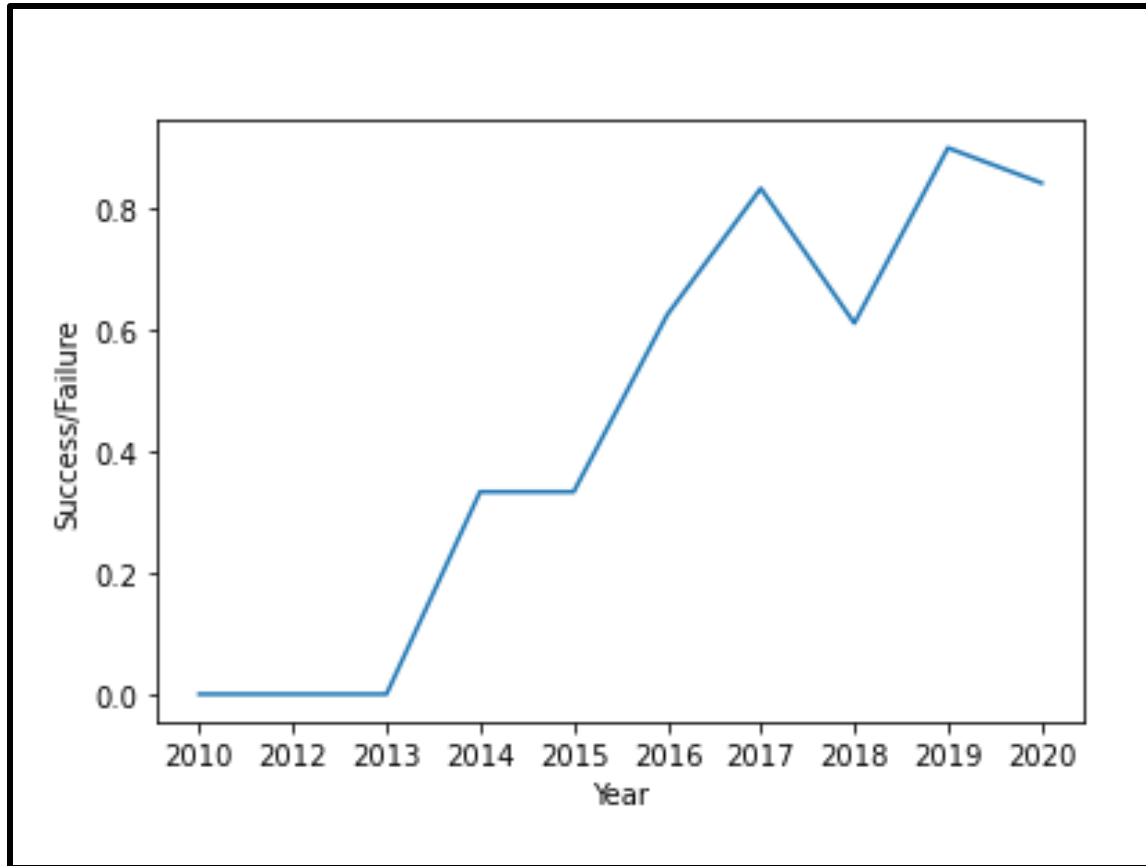
PAYLOAD BY ORBIT

Successful Landings by Orbit and Payload Mass (kg)



With heavy payloads the successful landing rate are more for PO, VLEO and ISS orbits. However, we cannot distinguish this for the GTO orbit.

SUCCESS OVER TIME



- Since 2013, SpaceX has generally observed an increase in annual mission success rates, indicating the company has been improving technology, learning from early failures, and may have reached a point of diminishing returns.
- This ability to achieve roughly 80% successful landing allows the reuse of rocket stage one, which in turn, allows SpaceX to significantly reduce its operating expenses

SQL: ALL LAUNCH SITE NAMES

- Use the keyword DISTINCT to show only unique launch sites from the SpaceX data

```
%%sql  
select distinct(LAUNCH_SITE) from SPACEXTBL;
```

Launch_Site

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

ALL LAUNCH SITES BEGINNING WITH 'CCA'



- We use the query below to display 5 records where launch sites begin with 'CCA'

```
%%sql
select * from SPACEXTBL where (LAUNCH_SITE) like 'CCA%' limit 5;
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

AVERAGE PAYLOAD MASS FOR F9 V1.1



- We use the query below to display the average payload mass carried by booster version F9 v1.1

```
%%sql
select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL
WHERE BOOSTER_VERSION LIKE 'F9 v1.1%';
```

payloadmass

2534.6666666666665

FIRST SUCCESSFUL GROUP PAD LANDING



- We use the query below to display the date of the first successful landing outcome in ground pad

```
%%sql
SELECT min(substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2)) as date_yyyymmdd FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)'
```

date_yyyymmdd

20151222 December 22, 2015

BOOSTERS WITH SUCCESS IN DRONE SHIP & PAYLOAD MASS BETWEEN 4000-6000 KG



- We use the query below to display the list of boosters which have success in drone ship and have a payload mass between 4,000 and 6,000 kg

```
%%sql SELECT booster_version FROM SPACEXTBL  
WHERE "Landing_Outcome" = 'Success (drone ship)'  
and payload_mass__kg_ between 4000 and 6000
```

Booster_Version

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

TOTAL SUCCESSFUL/FAILED MISSION OUTCOMES



- We use the query below to display a list of total successful or failed mission outcomes

```
%%sql select MISSION_OUTCOME, count(MISSION_OUTCOME) as SUM_OUTCOME
from SPACEXTBL group by MISSION_OUTCOME
```

Mission_Outcome	SUM_OUTCOME
Failure (in flight)	1
Success (payload status unclear)	1
Success	99

BOOSTERS CARRYING MAXIMUM PAYLOAD

- We use the query below to display the list of boosters versions which have carried the maximum payload mass

```
%%sql select BOOSTER_VERSION as boosterversion  
from SPACEXTBL where PAYLOAD_MASS_KG_=(select max(PAYLOAD_MASS_KG_) from SPACEXTBL);
```

boosterversion

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

BOOSTER & LAUNCH SITE IN FAILED DRONE SHIP LANDINGS IN 2015



- We use the query below to display the launch site, booster versions, and month during which there were failed drone ship landing outcomes in 2015.

```
%%sql
SELECT "Booster_Version", "Launch_Site" , substr(Date, 4, 2) as month FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Failure (drone ship)' and SUBSTR(Date,7,4)='2015'
```

Booster_Version	Launch_Site	month
F9 v1.1 B1012	CCAFS LC-40	01
F9 v1.1 B1015	CCAFS LC-40	04

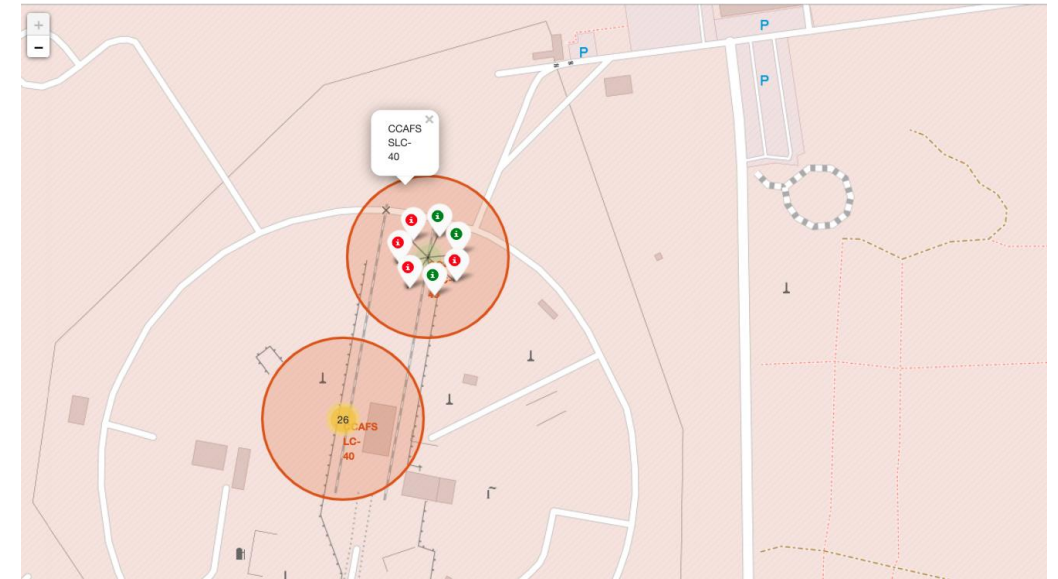
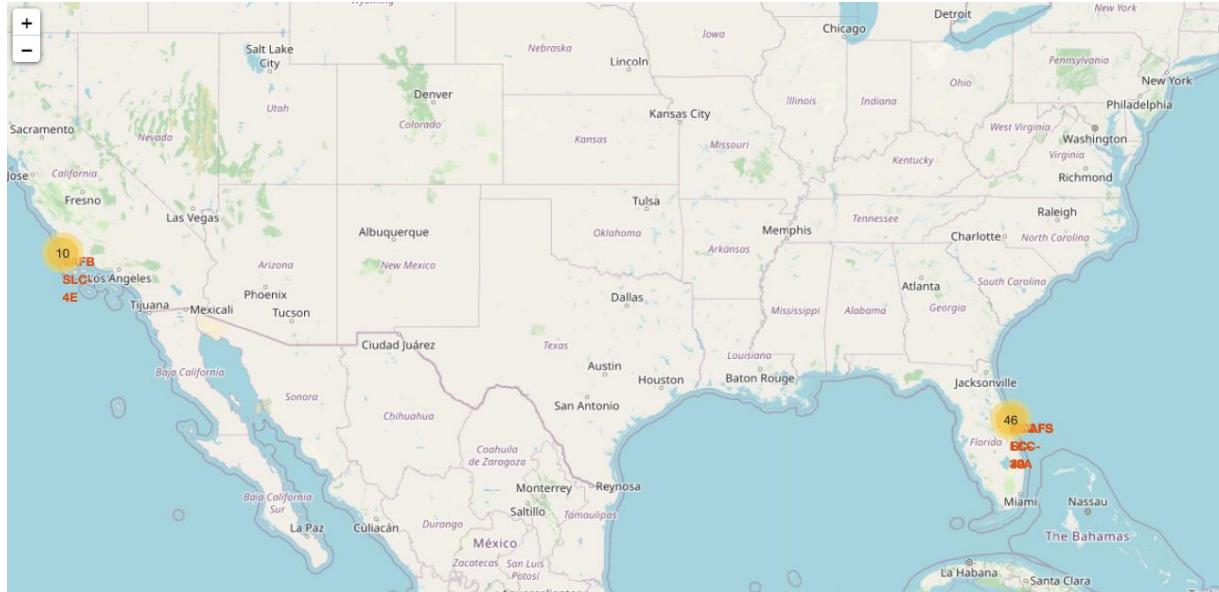
RANKED COUNT OF SUCCESSFUL LANDINGS

- We use the query below to display a ranked count of successful landing outcomes between 04-06-2014 and 20-03-2017 (in descending order).

```
%%sql
SELECT "Landing _Outcome", COUNT("Landing _Outcome")
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
AND "Landing _Outcome" like "%Success%"
ORDER BY COUNT("Landing _Outcome") DESC
```

landingoutcome	count
Success (drone ship)	6
Success (ground pad)	5
Controlled (ocean)	3

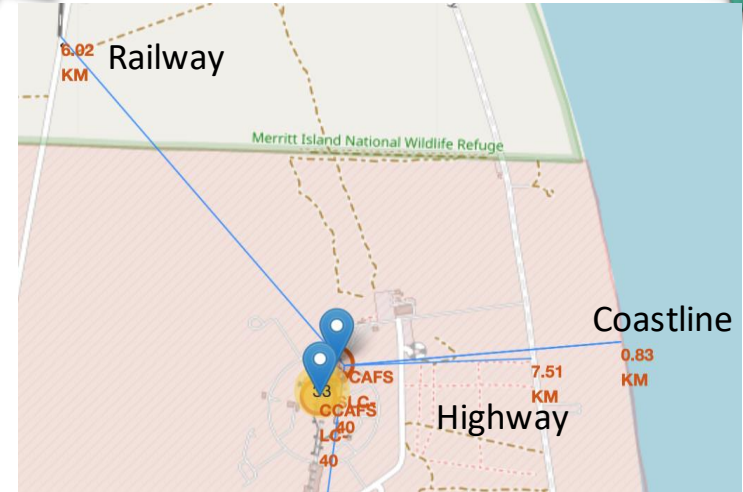
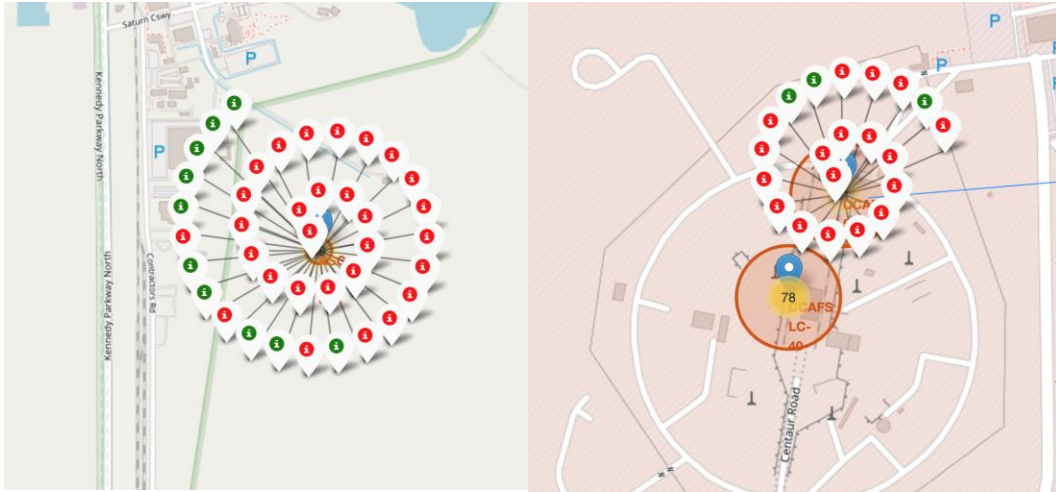
FOLIUM INTERACTIVE MAP



This interactive map allows us to observe:

- Launch sites are built *relatively* close to infrastructure such as highways & railways (<10 km) to facilitate of transportation of cargo, large machinery, and people to the launch site; however, the launch site is still located a safe distance away these public utilities.
- Additionally, they are built very close to the coastline (<10 km), presumably to allow for water landings and safer aborted missions. Logic suggests that, due to the rotation of the earth, having a rocket take off from the east coast over open water would permit any debris to fall into the ocean, should anything go wrong during takeoff.
- Finally, launch sites are located quite a distance (>50 km) from the nearest city, assuredly for safety reasons in case of emergency.

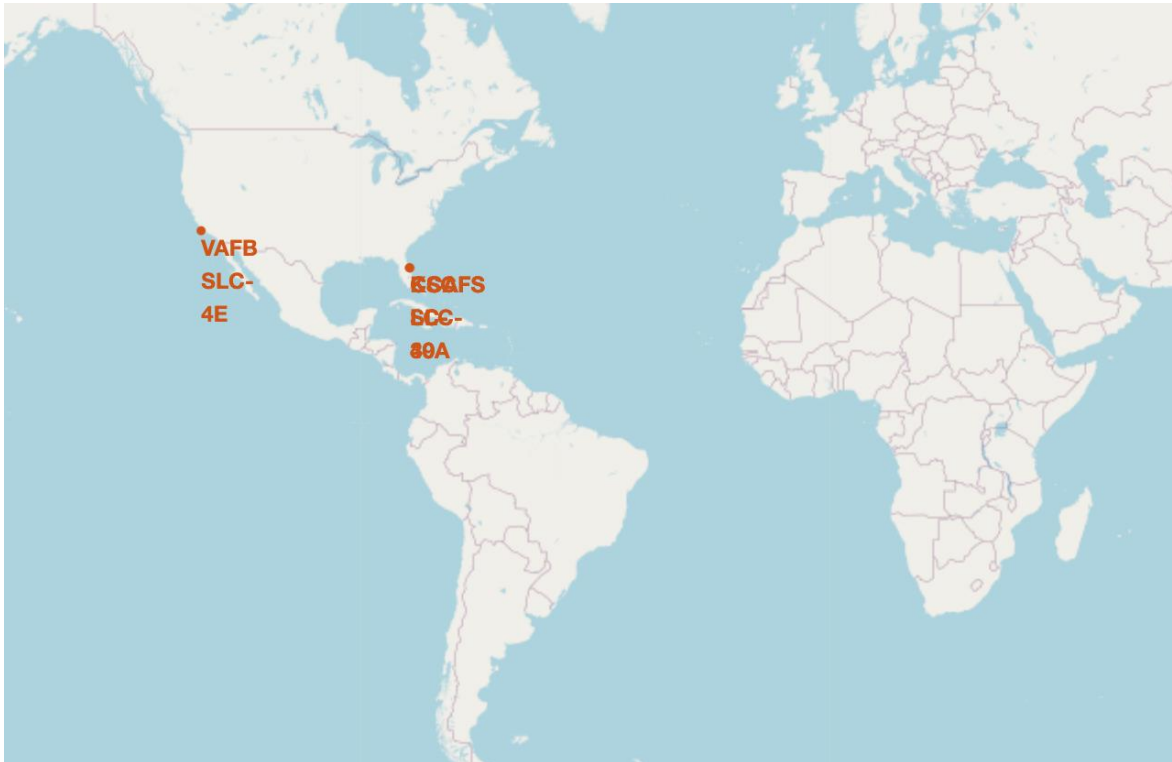
LAUNCH SITE PROXIMITY ANALYSIS



This interactive map allows us to observe:

- Launch sites are built *relatively* close to infrastructure such as highways & railways (<10 km) to facilitate of transportation of cargo, large machinery, and people to the launch site; however, the launch site is still located a safe distance away these public utilities.
- Additionally, they are built very close to the coastline (<10 km), presumably to allow for water landings and safer aborted missions. Logic suggests that, due to the rotation of the earth, having a rocket take off from the east coast over open water would permit any debris to fall into the ocean, should anything go wrong during takeoff.
- Finally, launch sites are located quite a distance (>50 km) from the nearest major city (Melbourne), assuredly for safety reasons in case of emergency.

LAUNCH SITE PROXIMITY ANALYSIS



- We use cluster markers to visualize at a glance how many launches have occurred at each site

- SpaceX launch sites are located on the east and west coast of the USA
- Three sites are located in Florida (KSC LC-39A, CCAFS SLC-40, CCAFS LC-40) and, one in California (VAFB SLC 4E).



PREDICTIVE ANALYTICS

```
models = {'KNeighbors': knn_cv.best_score_,
          'DecisionTree': tree_cv.best_score_,
          'LogisticRegression': logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

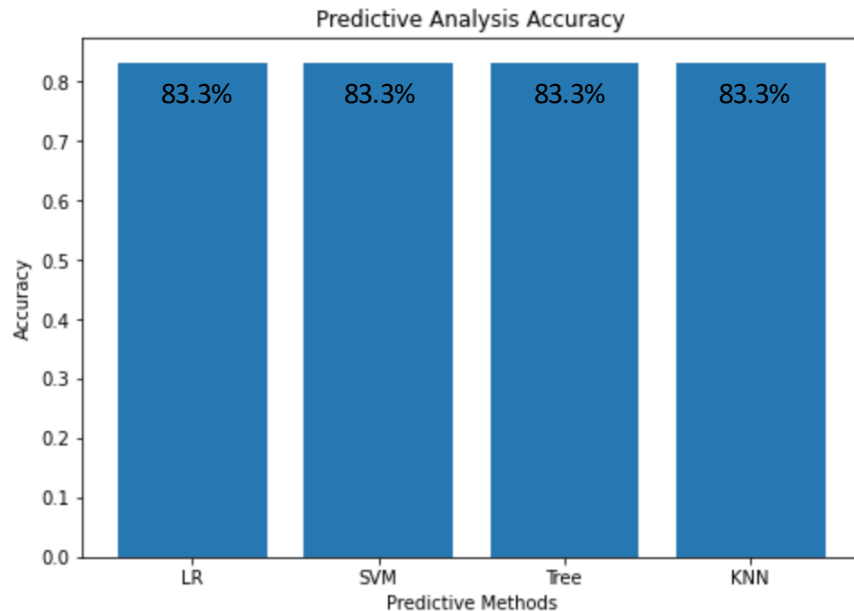
bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

Best model is DecisionTree with a score of 0.8732142857142856

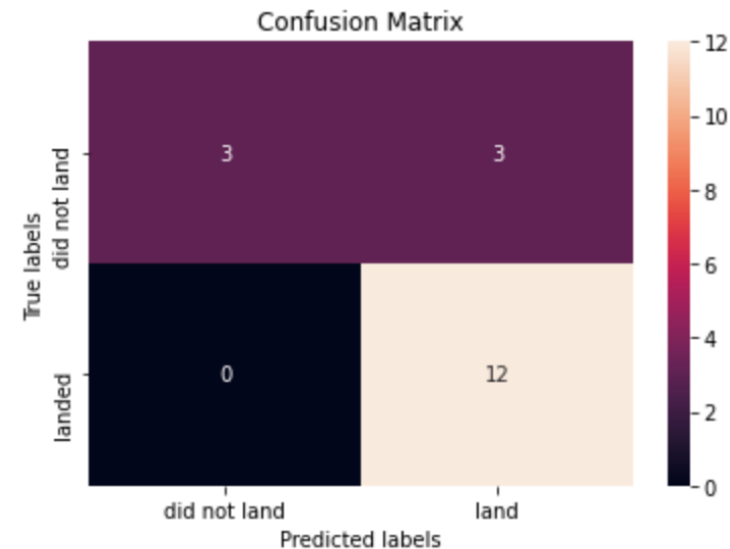
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}

- In training, the Decision Tree is the best classifier, correctly categorizing 87.3% of training data. However, the small training and sample sizes of our data limit the models' predictive ability.
- Due to the small number of launches, the training data consists of 72 entries and the test set consists of 18 entries. Thus, each of the models have equal predicted accuracy of 83.3%.

PREDICTIVE ANALYTICS



- Due to the small training/test size of the data, all of the models were equally predictive at 0.833.
- The confusion matrices correctly predict that 12 rockets that do land will land; however, they incorrectly predicted that three rockets would land successfully that fail to land (a significant false positive problem).



PREDICTIVE ANALYTICS

Logistic Regression

```
# define hyperparameters to tune
parameters_lr = {'C':[0.01,0.1,1],
                 'penalty':['l2'],
                 'solver':['lbfgs']}# l1 lasso l2 ridge

# define the model
lr = LogisticRegression(random_state = 12345)

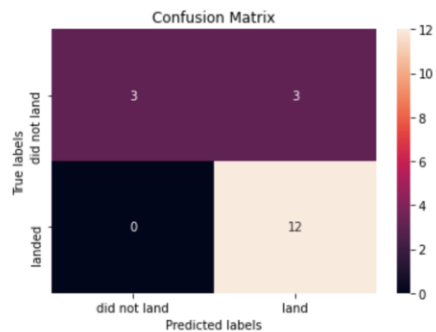
# define the grid search object
grid_search_lr = GridSearchCV(
    estimator = lr,
    param_grid = parameters_lr,
    scoring = 'accuracy',
    cv = 10
)

# execute search
logreg_cv = grid_search_lr.fit(X_train,Y_train)
```

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

```
tuned hyperparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8472222222222222
```

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



```
print('Accuracy of test data is: {:.3f}'.format(logreg_cv.score(X_test, Y_test)))
```

Accuracy of test data is: 0.833

SVM

```
# define hyperparameters to tune
parameters_svm = {'kernel':('linear','rbf','poly','rbf','sigmoid'),
                  'C': np.logspace(-3, 3, 5),
                  'gamma':np.logspace(-3, 3, 5)}

# define the model
svm = SVC(random_state = 12345)
```

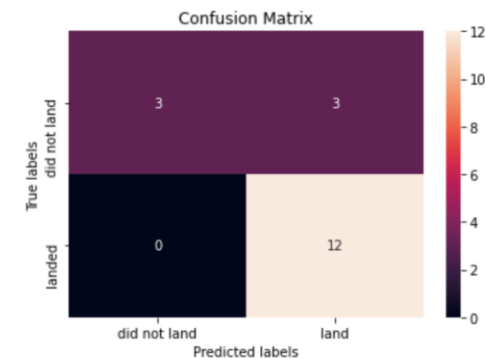
```
# define the grid search object
grid_search_svm = GridSearchCV(
    estimator = svm,
    param_grid = parameters_svm,
    scoring = 'accuracy',
    cv = 10
)
```

```
# execute search
svm_cv = grid_search_svm.fit(X_train,Y_train)
```

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

```
tuned hyperparameters :(best parameters) {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8472222222222222
```

```
yhat_svm = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_svm)
```



```
print('Accuracy of test data is: {:.3f}'.format(svm_cv.score(X_test, Y_test)))
```

Accuracy of test data is: 0.833

PREDICTIVE ANALYTICS

Decision Tree

```
# define hyperparameters to tune
parameters_tree = {'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]}

# define the model
tree = DecisionTreeClassifier(random_state = 12345)

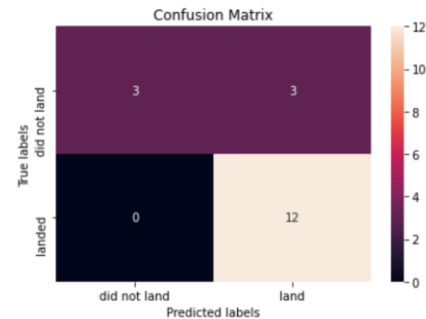
# define the grid search object
grid_search_tree = GridSearchCV(
    estimator = tree,
    param_grid = parameters_tree,
    scoring = 'accuracy',
    cv = 10
)

# execute search
tree_cv = grid_search_tree.fit(X_train, Y_train)

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

tuned hyperparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.875
```

```
yhat_tree = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat_tree)
```



```
print('Accuracy of test data is: {:.3f}'.format(tree_cv.score(X_test, Y_test)))
```

Accuracy of test data is: 0.833

KNN

```
# define hyperparameters to tune
parameters_knn = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                  'algorithm': ['auto', 'ball tree', 'kd tree', 'brute'],
                  'p': [1, 2]}

# define the model
knn = KNeighborsClassifier()

# define the grid search object
grid_search_knn = GridSearchCV(
    estimator = knn,
    param_grid = parameters_knn,
    scoring = 'accuracy',
    cv = 10
)

# execute search
knn_cv = grid_search_knn.fit(X_train, Y_train)

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

tuned hyperparameters :(best parameters) {'algorithm': 'auto', 'n_neighbors': 9, 'p': 1}
accuracy : 0.8472222222222222

yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test, yhat)
```



```
print('Accuracy on test data is: {:.3f}'.format(knn_cv.score(X_test, Y_test)))
```

Accuracy on test data is: 0.833

DISCUSSION

Findings

- KSC LC-39A has had the most successful launches from all sites
- Orbits GEO, HEO, SSO, & ES-L1 have the best success rate, but are infrequently used; VLEO is the orbit with the most usage and highest success rate.
- Low-weighted payloads perform better than heavier payloads.
- Success rates for SpaceX launches is correlated with time: later launches are more successful.

Implications

- Launch Site Matters
- Orbit Matters
- Pay Load Matters
- They're getting better with time



CONCLUSIONS

- It is worth noting that the small sample size of our data limits the predictive capabilities of our models. However, we can say the following:
- The SVM, KNN, and Logistic Regression models are best in terms of prediction accuracy for this dataset.
- Low weighted payloads perform better than heavier payloads.
- The success rates of SpaceX launches is proportional to the time in years.
- KSC LC 39A has had the most successful launches of all sites.
- Orbits GEO, HEO, SSO, ES-L1 have the best success rates.

APPENDIX

"List of Falcon 9 and Falcon Heavy launches" Wikimedia Foundation, 13 June, 2022 https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches