Winning Space Race with Data Science

Harald Penasso
October 18, 2021

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

Data collection – API

- The SpaceX application programming interface (API) and a JSON-file static response object are used to retrieve and and extract rocket launch data into a pandas dataframe.

Data collection – Web Scraping

- Retrieve the content of the SpaceX Wikipedia page and extract its tables.

Data wrangling

- The dataframe is reduced to features representing: "**rocket**", one "**payload**" only, "**launchpad**", one "**core**" only, "**flight number**", and "**date**" which are further split into more distinct features.
- Falcon 1 launches are removed.
- Missing payload mass values are replaced with the column mean.
- Perform one-hot-encoding on categorical data

Executive Summary

Exploratory data analysis

- Inspect the features ‘Launch pads’ and ‘Orbits’
- Seaborn – Visualize features on x- and y-axis together with the landing outcome in color and show first stage success rate for each orbit and over time
- SQL – Upload the data to IBM DB2 server and use SQL magic in a Jupyter notebook to extract information.

Interactive visual analytics

- Folium – Mark launch sites on map and use marker clusters to represent success rate. Calculate and display distances between launch site(s) and its proximities.
- Plotly Dash – Select all or specific launch sites and limit payload range to visualize success rate and its relation to payload mass and booster version with a pie-chart as well as on a scatter plot.

Executive Summary

Predictive analysis using classification models

- After standardizing and splitting the data into train and test sets, we use a hyperparameter grid search to find the best classifier to predict first stage landing.
- Classifiers used:
  - Logistic Regression
  - Support Vector Machine
  - Decision Tree
  - k-Nearest Neighbors

Executive Summary

Results

- The mission outcome is rated 98% successful, and two thirds of Falcon 9 first stages land successfully
- Most successful stage 1 landings are on drone ships
- Targeted orbits and payloads have different stage 1 landing success rates
- Success rate improves over time
- Launch sites are near the coast lines and transport infrastructure but avoid cities.
- Launch site KSC LC-39A and the FT booster version are the most successful
- Classifiers have similar performance

Introduction

The increasing demand for the deployment of satellites in Earth's orbit, for scientific missions, and recently space travel opened a new space race: the reduction of launch costs.

While the first stage of a rocket performs the major lifting work, disconnects from stage two and the cargo after main engine cutoff, and is commonly destroyed after that, it is also one of the most expensive parts of the rocket.

Safely landing and re-using fist stage rockets therefore greatly reduces mission costs.

In order to estimate competitive future launch prices, we aim to predict SpaceX first stage landing success from previous launches.

Section 1

Methodology

Data Collection – SpaceX API

```
import requests
import pandas as pd
import numpy as np
import datetime
```

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

```
response = requests.get(spacex_url)
```

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

```
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

```
data = data[data['cores'].map(len)==1]
```

```
data = data[data['payloads'].map(len)==1]
```

```
data['cores'] = data['cores'].map(lambda x : x[0])
```

```
data['payloads'] = data['payloads'].map(lambda x : x[0])
```

```
data['date'] = pd.to_datetime(data['date_utc']).dt.date
```

```
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Extract features BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude using provided functions accessing the related SpaceX APIs.

```
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}
```

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

```
data_falcon9 = launch_df[launch_df['BoosterVersion']!='Falcon 1']
```

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

```
PayloadMass_mn = data_falcon9['PayloadMass'].mean()
```

```
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan,PayloadMass_mn)
```

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/01%20Data%20Collection%20API%20Lab.ipynb

Data Collection – Scraping

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

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

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

```
falcon_wiki_soup = BeautifulSoup(falcon_wiki, "html.parser")
```

```
html_tables = falcon_wiki_soup.find_all('table')
```

```
column_names = []
for row in first_launch_table.find_all('th'):
    content = extract_column_from_header(row)
    if content is not None and len(content) > 0:
        column_names.append(content)
```

```
launch_dict= dict.fromkeys(column_names)
```

```
del launch_dict['Date and time ( )']
```

```
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Data Collection – Scraping

```
extracted_row = 0
for table_number,table in enumerate(falcon_wiki_soup.find_all('table',"wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        row=rows.find_all('td')
        if flag:
            extracted_row += 1
            # Flight Number value
            launch_dict['Flight No.'].append(flight_number)
            # Date value
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            # Time value
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            # Booster version
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            # Launch Site
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            # Payload
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            # Payload Mass
            payload_mass = get_mass(row[4])
            launch_dict['Payload mass'].append(payload_mass)
            # Orbit
            orbit = row[5].a.string
            launch_dict['Orbit'].append(orbit)
            # Customer
            if row[6].a is not None and len(row[6].a) > 0:
                customer = row[6].a.string
            launch_dict['Customer'].append(customer)
            # Launch outcome
            launch_outcome = list(row[7].strings)[0]
            ['Launch outcome'].append(launch_outcome)
            # Booster landing
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
```

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/02%20Data%20Collection%20with%20Web%20Scraping.ipynb

Data Wrangling

From the Pandas dataframe the target feature ‘Launch outcome’ was encoded to to a binary format in column ‘class’:

0 : First stage **did not** land successfully

1 : First stage **did** land successfully

Data Wrangling

```
import pandas as pd
import numpy as np
```

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

```
df.isnull().sum()/df.count()*100
```

```
df.LaunchSite.value_counts()
```

```
df.Orbit.value_counts()
```

```
landing_outcomes = df.Outcome.value_counts()
```

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

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

```
landing_class = np.int64(df.Outcome.isin(bad_outcomes).values==False).tolist()
```

```
df['Class']=landing_class
```

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

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/03%20EDA.ipynb

EDA with Data Visualization

- The seaborn package function catplot was used to create scatter plots for continuous and categorical data.
  - It allows to easily color code a 3rd dimension on the 2D plot, i.e., Falcon 9 first stage landing success and supports direct datagram data extraction via column names.
- Line and bar charts are directly plotted via data from objects which embed matplotlib as part of the Pandas package

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/05%20EDA%20with%20Visualization.ipynb

EDA with SQL

• SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXDATASET	Show Unique Launch Site names
• SELECT * FROM SPACEXDATASET WHERE LAUNCH_SITE LIKE '%CCA%' LIMIT 5	Five Launch Sites having *CCA* in their name
• SELECT SUM(payload_mass__kg_) FROM SPACEXDATASET WHERE CUSTOMER LIKE '%NASA%‘	The total payload mass transported for NASA
• SELECT AVG(payload_mass__kg_) FROM SPACEXDATASET WHERE BOOSTER_VERSION LIKE '%F9 v1.1%‘	Average payload mass carried by Booster F9 v1.1
• SELECT MIN(DATE) FROM SPACEXDATASET WHERE landing__outcome LIKE '%Success (ground pad)%‘	First successful landing on a ground pad
• SELECT BOOSTER_VERSION FROM SPACEXDATASET WHERE landing__outcome LIKE '%Success (drone ship)%' AND payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000	Booster versions carrying payloads between 4 and 6 tons successfully landing on drone ship

EDA with SQL

- SELECT mission_outcome, COUNT(*) as RATE FROM SPACEXDATASET GROUP BY mission_outcome

  Number of successful mission outcome per category

- SELECT booster_version FROM SPACEXDATASET
  WHERE payload_mass__kg_ = (
  SELECT MAX(payload_mass__kg_) FROM SPACEXDATASET
)

  Boosters lifting the heaviest payloads

- SELECT landing__outcome, BOOSTER_VERSION, launch_site
  FROM SPACEXDATASET
  WHERE YEAR(DATE) = 2015 AND landing__outcome LIKE 'Fail%‘

  Landing fails in 2015 showing booster and launch site

- SELECT landing__outcome, COUNT(*) AS "Count" FROM SPACEXDATASET
  WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20‘
  GROUP BY landing__outcome
  ORDER BY "Count“
  DESC

  Order the count of successful landing types between dates

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/04%20EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- An OpenStreetMap was displayed via Folium
- Circles were added to the map to highlight the locations of launch sites
- Markers were aligned with those circles to give them meaning
- Markers were grouped by marker clusters to visualize success rate as color-coded icons
- Lines (`PolyLine`) were drawn from launch site coordinates to points of interest. The distances measured were printed onto the map next to the line. Point of interest coordinates were extracted from mouse positions.

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/06%20Interactive%20Visual%20Analytics%20with%20Folium.ipynb

https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/70a7263d-286e-486e-8c6a-807aa6d9fa05/view?access_token=a199ad00c954d0051be418177637fc74badfac01ec25d75d3fba198b7e4ddee3

Build a Dashboard with Plotly Dash

- A pie chart and a scatter plot were added
  (I used the Plotly `strip` to enable jitter and to avoid overlayed, stacked, hidden datapoints).
- The dropdown menu selects the filter option allowing the user to either display all or only specific launch site data on the plots.
- The range slider allows selective plot rendering based on the payload mass selection.
- The pie chart gives the user a quick impression on an individual launch site success rate – or shows where the most successfully landed first stages had started.
- The scatter plot explores the relation between first stage landing success, payload and booster type.

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/07%20SpaceX%20Dash%20App.py

Predictive Analysis (Classification)

- After standardizing and splitting the data into train and test sets, we used hyperparameter grid search to find the best classifier to predict first stage landing.
- Classifiers tested: Logistic Regression, Support Vector Machine, Decision Tree, and k-Nearest Neighbors.
- Each classifier was trained using the the training set, specified grid search hyperparameters were used to find their best performing combination regarding the classifier prediction on the test set.
- Accuracy, F1 score, and confusion matrices were calculated and displayed to quantify classification model performance.

Predictive Analysis (Classification)

IMPORT

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

DATA PREPERATON

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

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

```
Y = data.Class.to_numpy()
```

```
X = preprocessing.StandardScaler().fit(X).transform(X)
```

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

Predictive Analysis (Classification)

LOGISTIC REGRESSION

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

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

```
print("best logreg parameter's score: ",logreg_cv.best_estimator_.score(X_test,Y_test))
```

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

SUPPORT VECTOR MACHING

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
svm_cv = GridSearchCV(estimator = svm, param_grid=parameters,cv=10)
svm_cv.fit(X_train,Y_train)
```

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

```
print("best svm parameter's score: ",svm_cv.best_estimator_.score(X_test,Y_test))
```

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

Predictive Analysis (Classification)

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

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

```
print("best decision tree parameter's score: ",tree_cv.best_estimator_.score(X_test,Y_test))
```

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

K-NEAREST NEIGHBOR

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

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

```
print("best knn parameter's score: ",knn_cv.best_estimator_.score(X_test,Y_test))
```

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

Predictive Analysis (Classification)

```
from sklearn.metrics import f1_score
plt.rcParams['figure.figsize'] = [10, 10]
fig = plt.figure()
yhat_logreg = logreg_cv.best_estimator_.predict(X_test)
f1_logreg = f1_score(Y_test,yhat_logreg)
acc_logreg = logreg_cv.best_estimator_.score(X_test,Y_test)
ax1 = plot_confusion_matrix(Y_test,yhat_logreg,221)
ax1.set_title('Best LogReg, F1 score: '+str(np.round(f1_logreg,3))+' Acc:
'+str(np.round(acc_logreg,4)*100)+'%' )

yhat_svm = svm_cv.best_estimator_.predict(X_test)
f1_svm = f1_score(Y_test,yhat_svm)
acc_svm = svm_cv.best_estimator_.score(X_test,Y_test)
ax2 = plot_confusion_matrix(Y_test,yhat_svm,222)
ax2.set_title('Best SVM, F1 score: '+str(np.round(f1_svm,3))+' Acc:
'+str(np.round(acc_svm,4)*100)+'%' )

yhat_tree = tree_cv.best_estimator_.predict(X_test)
f1_tree = f1_score(Y_test,yhat_tree)
acc_tree = tree_cv.best_estimator_.score(X_test,Y_test)
ax3 = plot_confusion_matrix(Y_test,yhat_tree,223)
ax3.set_title('Best DecTree, F1 score: '+str(np.round(f1_tree,3))+' Acc:
'+str(np.round(acc_tree,4)*100)+'%' )

yhat_KNN = knn_cv.best_estimator_.predict(X_test)
f1_KNN = f1_score(Y_test,yhat_KNN)
acc_KNN = knn_cv.best_estimator_.score(X_test,Y_test)
ax4 = plot_confusion_matrix(Y_test,yhat_KNN,224)
ax4.set_title('Best KNN, F1 score: '+str(np.round(f1_KNN,3))+' Acc:
'+str(np.round(acc_KNN,4)*100)+'%' )
```

RE-TRAINING AND EVALUATING THE BEST PERFORMING CLASSIFIERT VERSIONS

https://github.com/haripen/Applied_Data_Science_Capstone/blob/33c41e119d5a89be1d226b2517c53c35e3e2d9c2/08%20Machine%20Learning%20Prediction.ipynb

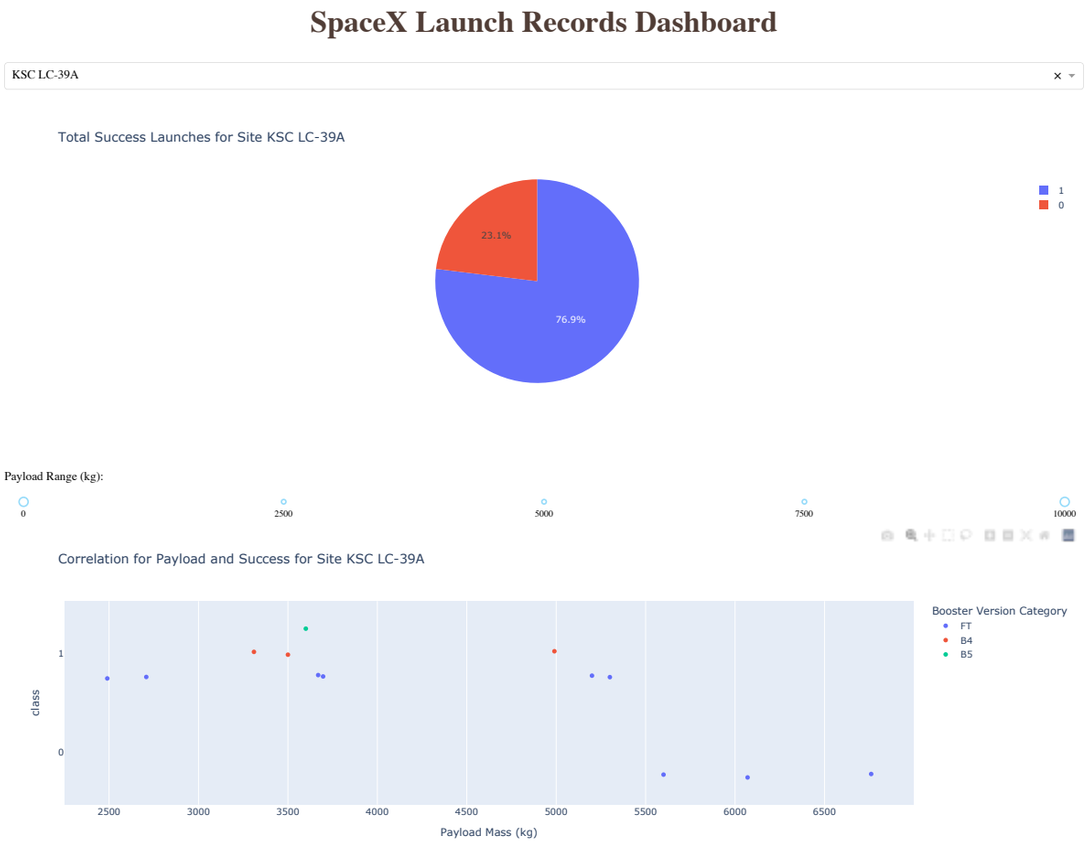
Results

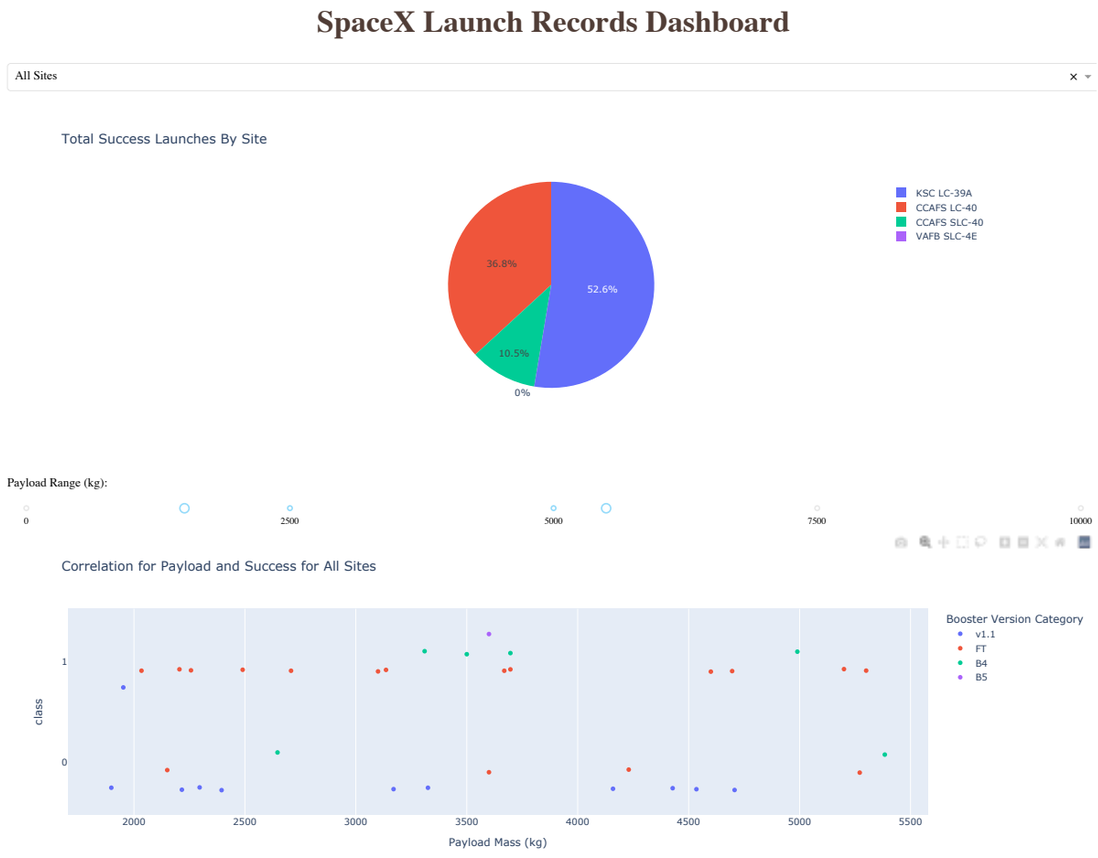
Exploratory data analysis results

- **Launch sites (number of launces):**
  CCAFS (S)LC 40 (55), KSC LC 39A (22) VAFB SLC 4E (13)
- **Orbits (number of missions to orbit)**
  GTO (27), ISS (21), VLEO (14), PO (9), LEO (7), SSO (5), MEO (3), HEO (1), ES-L1 (1), GEO (1), SO (1)
- **Stage one landing success rate:** 66.6%
- A total payload of 107.010 tons was lifted for NASA
- The F9 v1.1 booster carries 2534 kg payload on average
- The first successful land landing of the Falcon F9 first stage was on 22-12-2015
- 99 of 101 SpaceX missions are successful

Results

Interactive analytics demo in screenshots

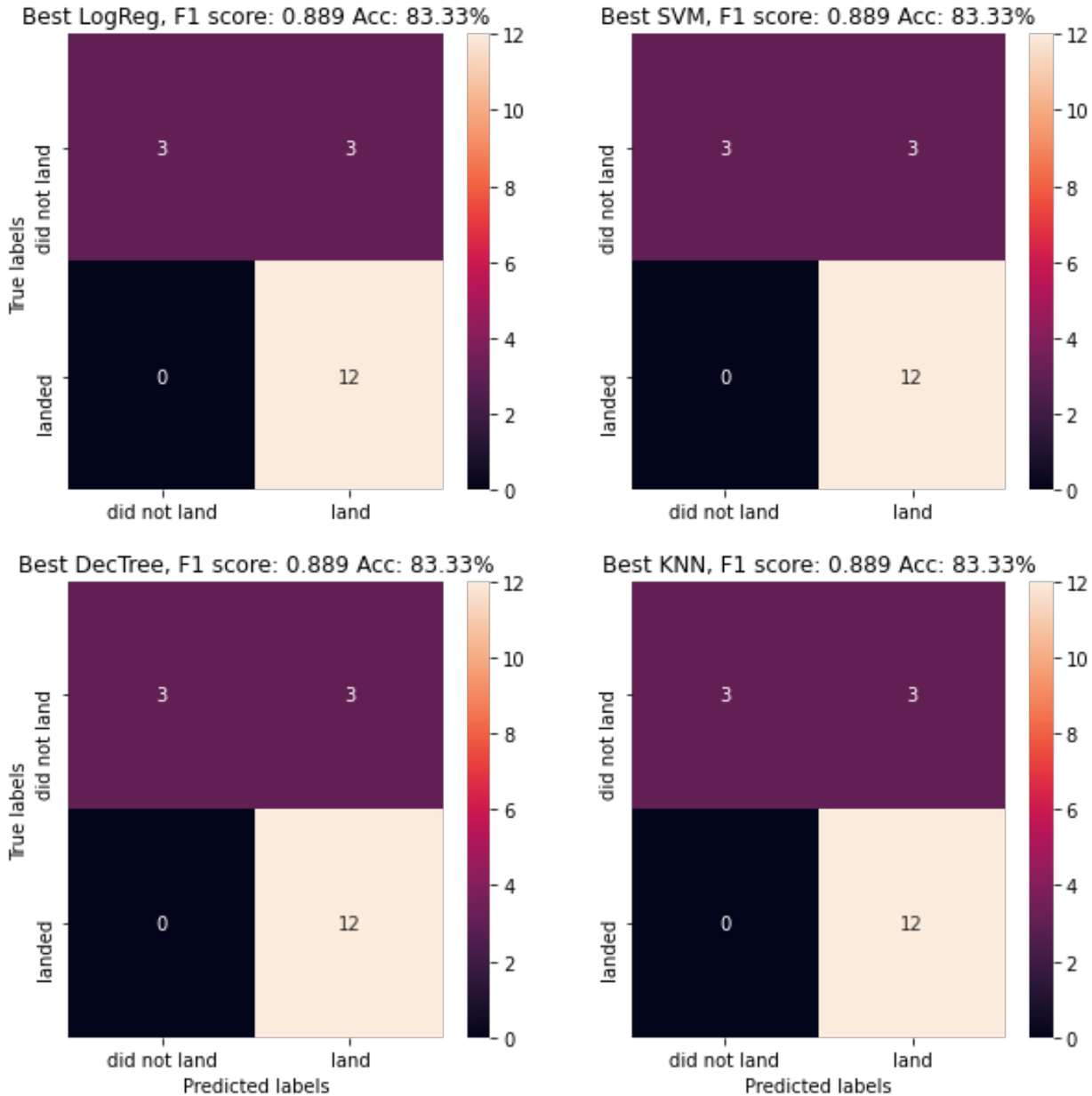




Results

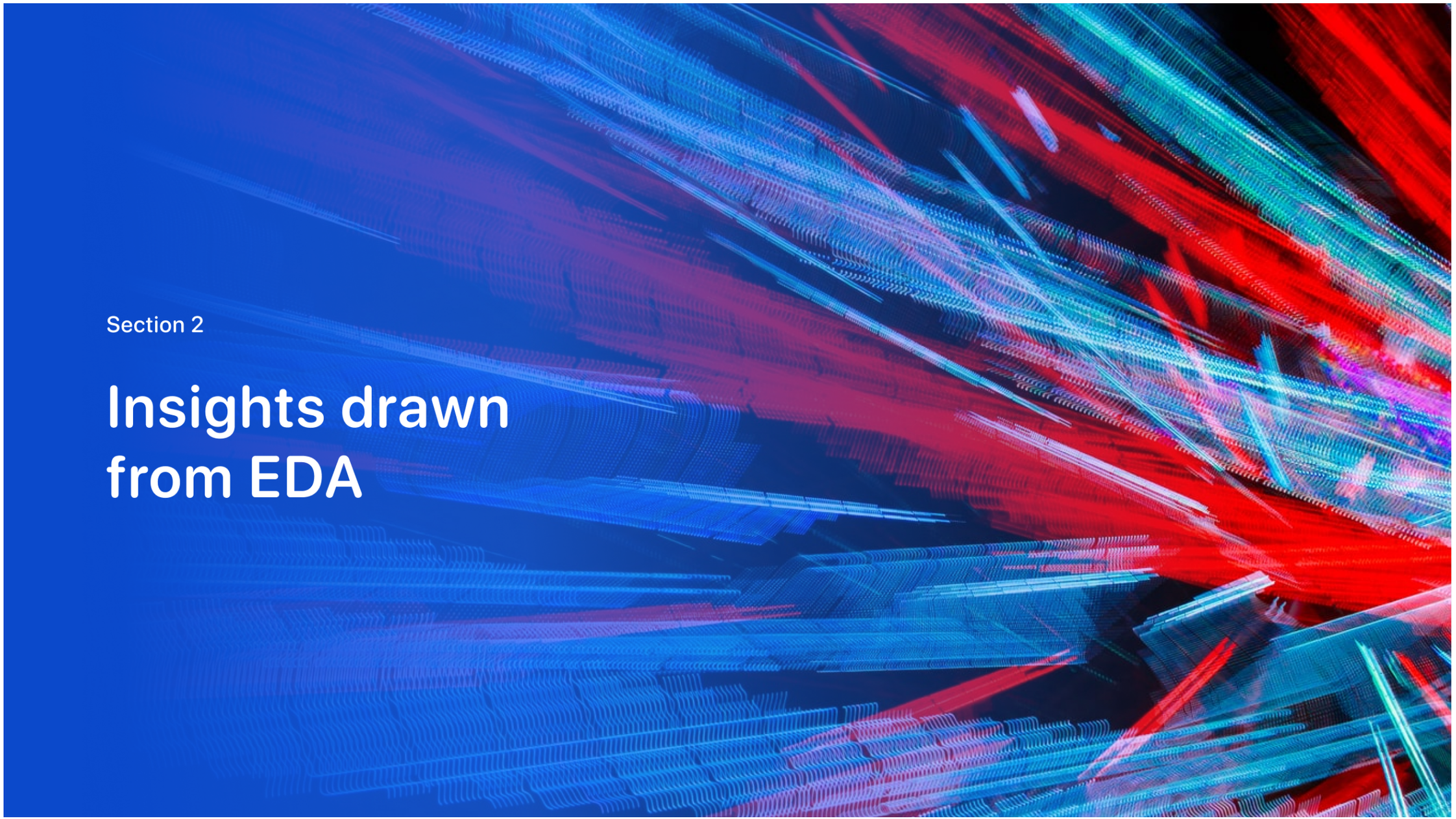
Predictive analysis results for the best performing classifier models (via hyperparameter tuning):

- F1 score
- Accuracy
- Confusion Matrix

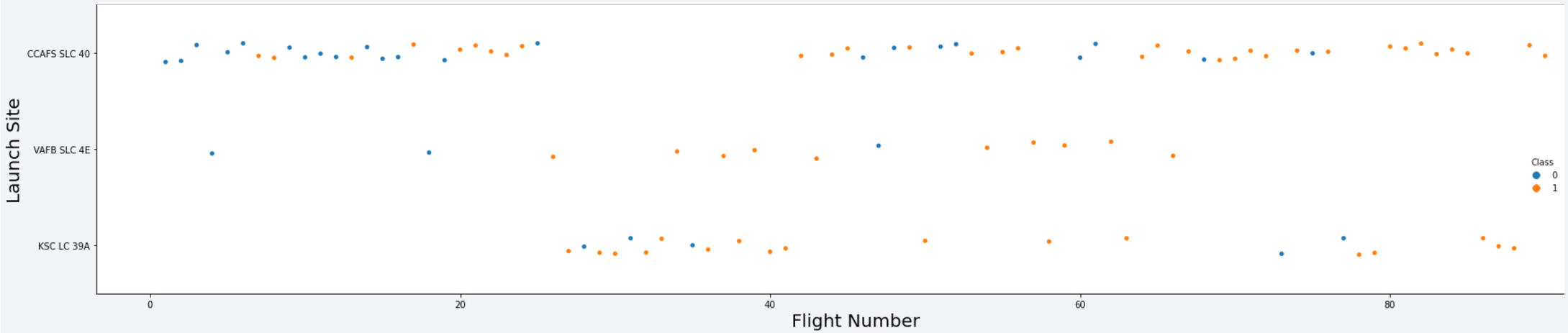


Section 2

Insights drawn from EDA

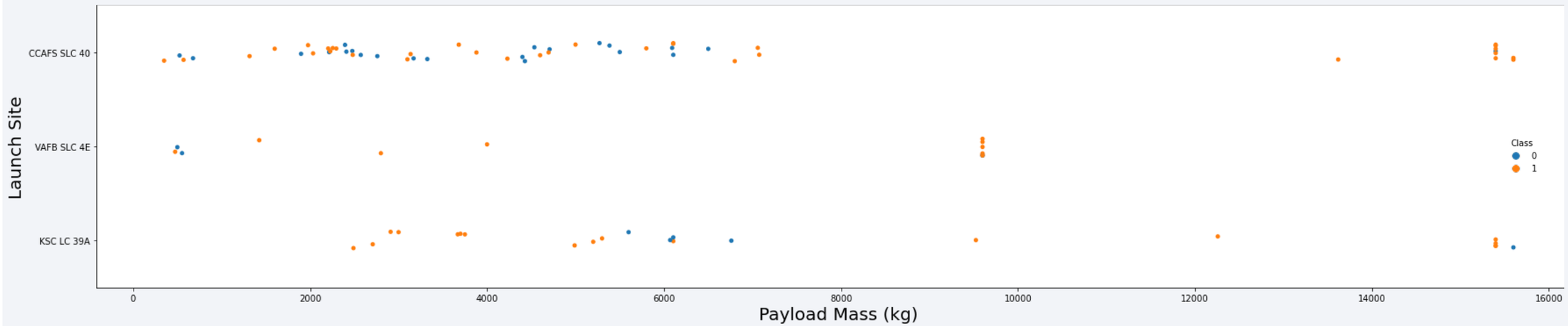


Flight Number vs. Launch Site



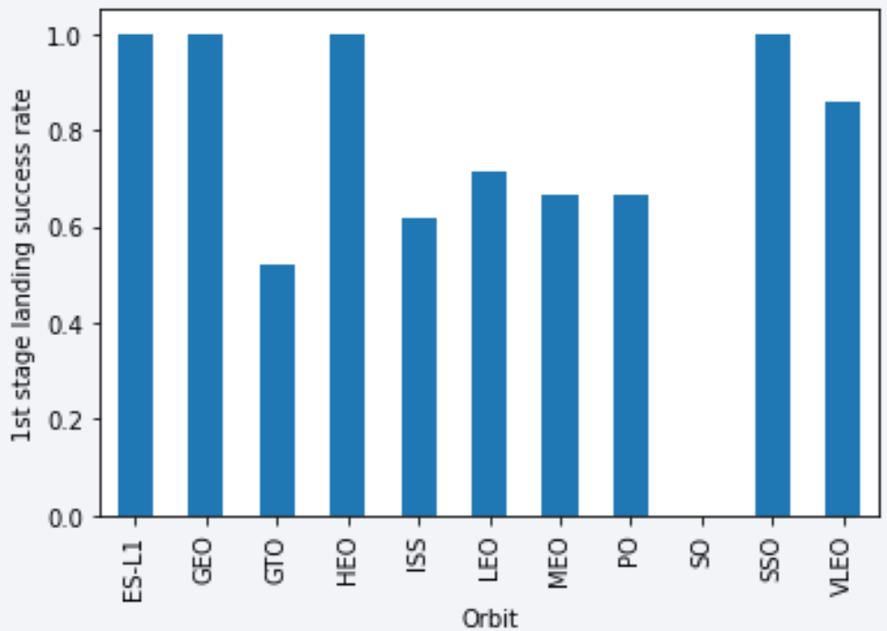
The most recently introduced launch site KSC LC 39A has the highest success rate.

Payload vs. Launch Site



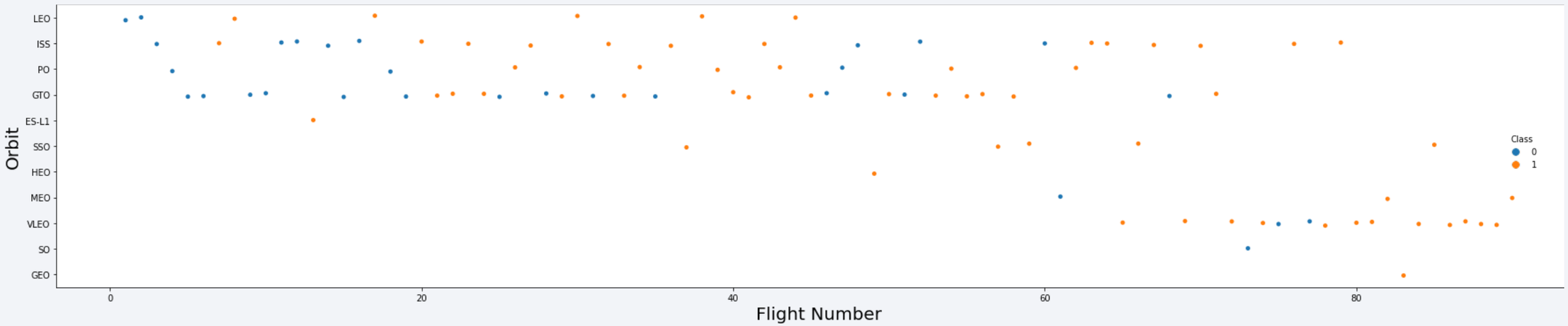
Launch site VAB SCL 4E did not handle any high payloads.

Success Rate vs. Orbit Type



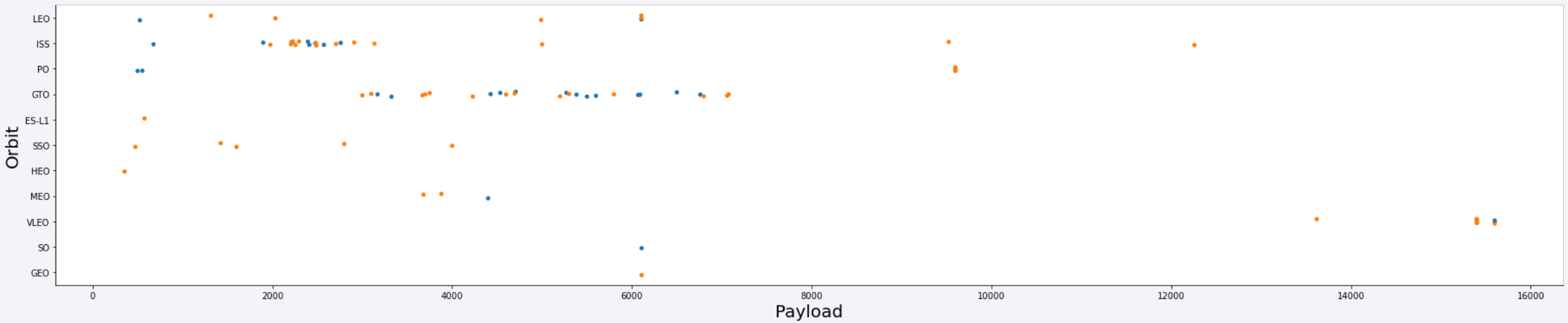
Some orbits have higher success rates for landing the first stage than others.

Flight Number vs. Orbit Type



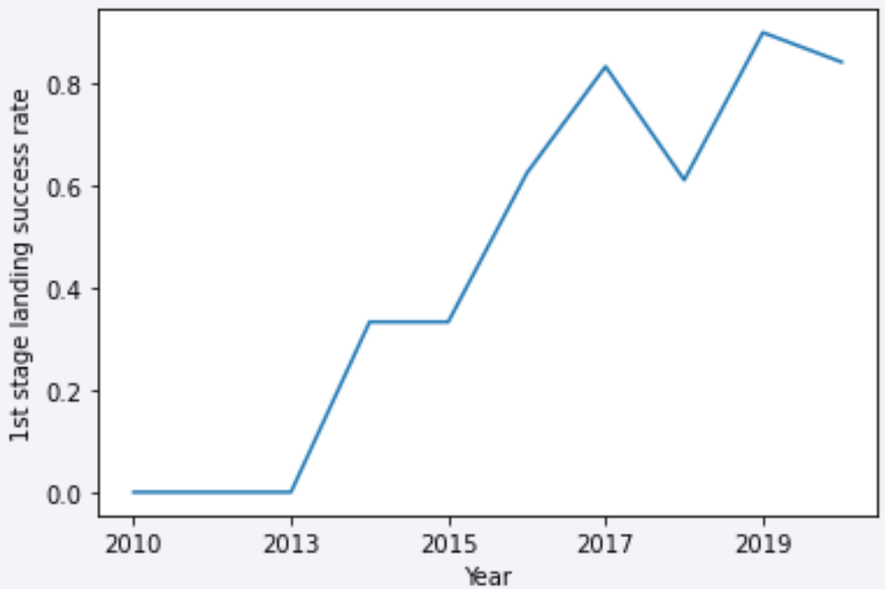
Most recent missions went to the very low Earth orbit (VLEO) with a high success rate.

Payload vs. Orbit Type



Some orbits, i.e., GTO, are associated with distinct payload ranges.

Launch Success Yearly Trend



The yearly average success rate for landing the Falcon 9 first stage improved over time.

All Launch Site Names

SQL command:
`SELECT UNIQUE(LAUNCH_SITE) FROM SPACEXDATASET`
returns the four launch site abbreviations

- Cape Canaveral Space Force Station:
  - Launch Complex 40: **CCAFS LC-40**
  - Space Launch Complex 40: **CCAFS SLC-40**
- Kennedy Space Center Launch Complex 39A: **KSC LC-39A**
- Vandenberg Space Launch Complex 4: **VAFB SLC-4E**

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

Launch Site Names Begin with 'CCA'

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

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

```
SELECT SUM(payload_mass__kg_) FROM SPACEXDATASET
  WHERE CUSTOMER LIKE '%NASA%‘
```

The total payload mass lifted for NASA is 107010 kg

Average Payload Mass by F9 v1.1

```
SELECT AVG(payload_mass__kg_) FROM SPACEXDATASET WHERE
  BOOSTER_VERSION LIKE '%F9 v1.1%‘
```

The average payload mass for the F9 v.1.1 booster is 2534 kg

First Successful Ground Landing Date

```
SELECT MIN(DATE) FROM SPACEXDATASET
  WHERE landing__outcome LIKE '%Success (ground pad)%‘
```

The first successful land-based landing was on 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
SELECT BOOSTER_VERSION FROM SPACEXDATASET
  WHERE landing__outcome LIKE '%Success (drone ship)%' AND
    payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000
```

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

Total Number of Successful and Failure Mission Outcomes

```
SELECT mission_outcome, COUNT(*) as RATE from SPACEXDATASET
  GROUP BY mission_outcome
```

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

Boosters Carried Maximum Payload

```
SELECT booster_version FROM SPACEXDATASET
  WHERE payload_mass__kg_ = ( SELECT MAX(payload_mass__kg_)
                                FROM SPACEXDATASET   )
```

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

2015 Launch Records

```
SELECT landing__outcome, BOOSTER_VERSION, launch_site FROM SPACEXDATASET
  WHERE YEAR(DATE)=2015 AND
  landing__outcome LIKE 'Fail%'
```

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

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

```
SELECT landing__outcome, COUNT(*) AS "Count" FROM SPACEXDATASET
  WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20‘
  GROUP BY landing__outcome
  ORDER BY "Count“
  DESC
```

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

Section 4

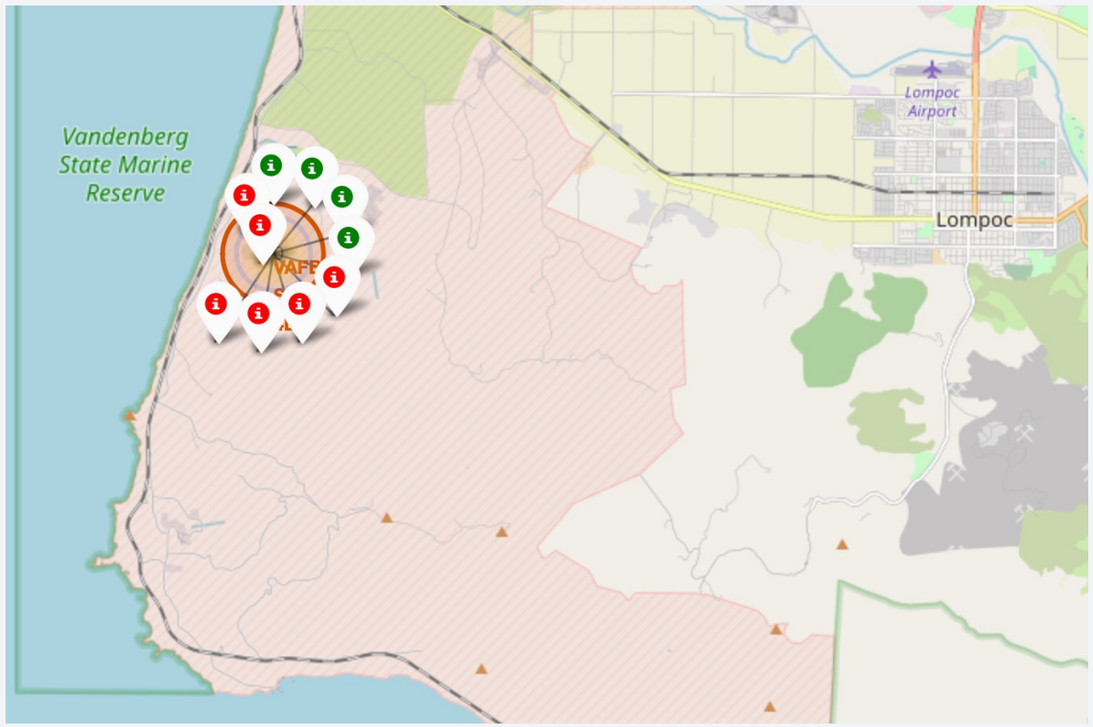
Launch Sites Proximities Analysis

SpaceX USA Launch Sites

The SpaceX launch site are in Florida (KSC LC-39A, CCAFS SLC 40, CCAFSSLC 40) and in California (VAFB SLC 4E).

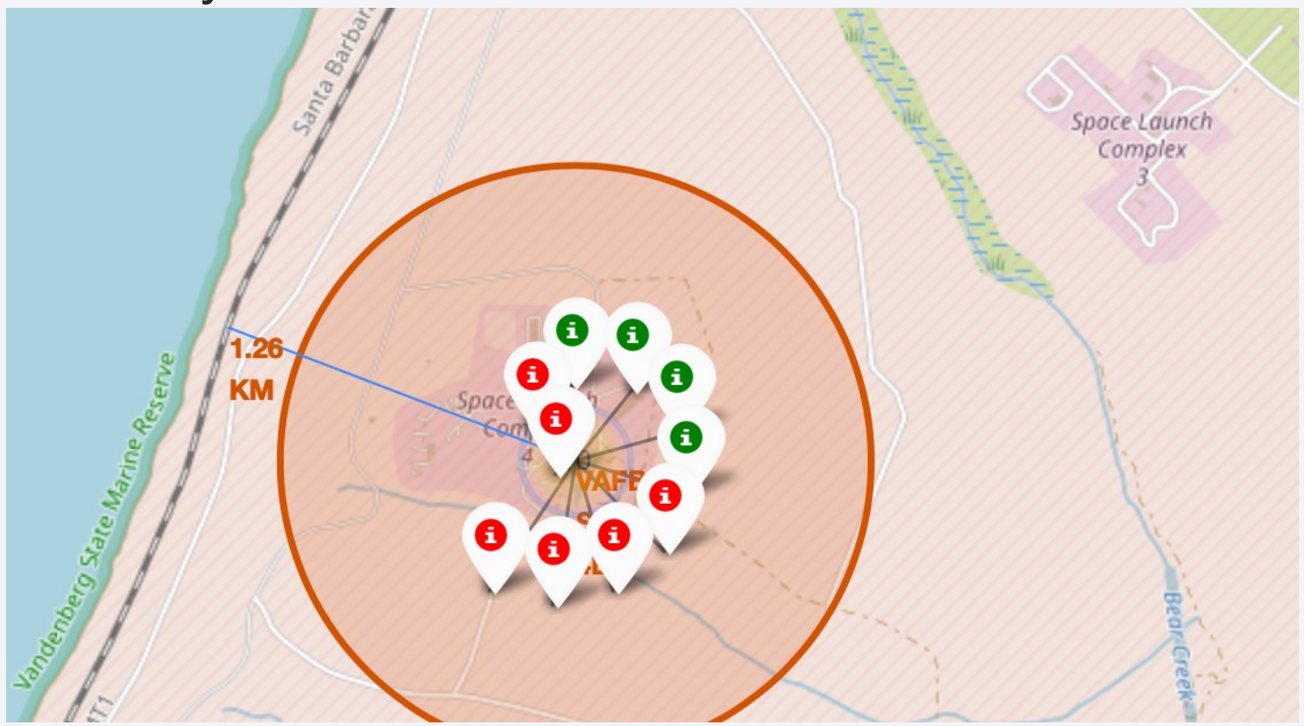
Vandenberg VAFB SLC 4E success rate

Only four launches from ten from VAFB SLC 4E resulted in successful landings of the Falcon F9 stage one.



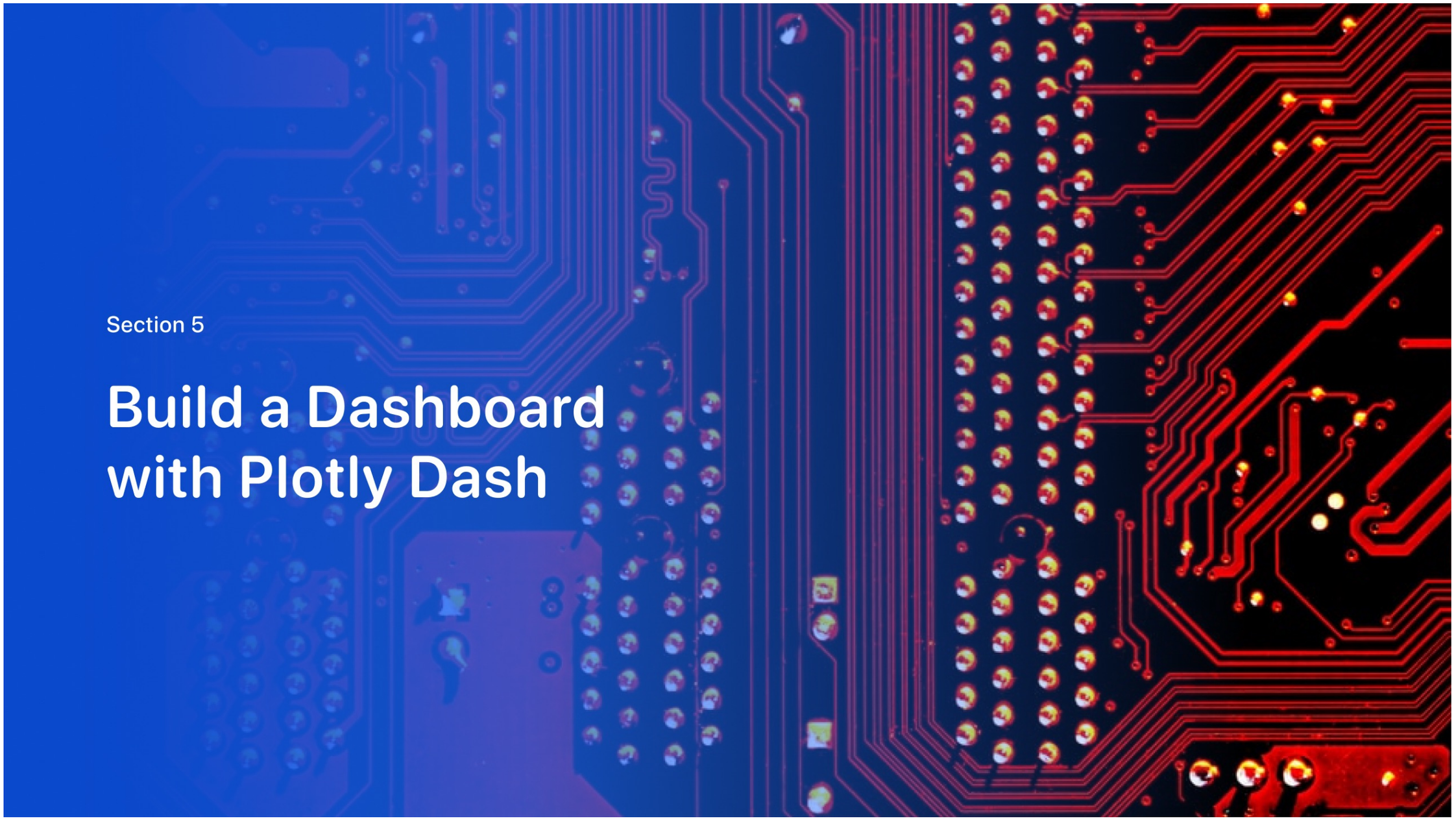
Railway near launch site VAFB SLC 4E

1.26 km is the distance between the VAFB SLC 4E launch site and the next railway track.



Section 5

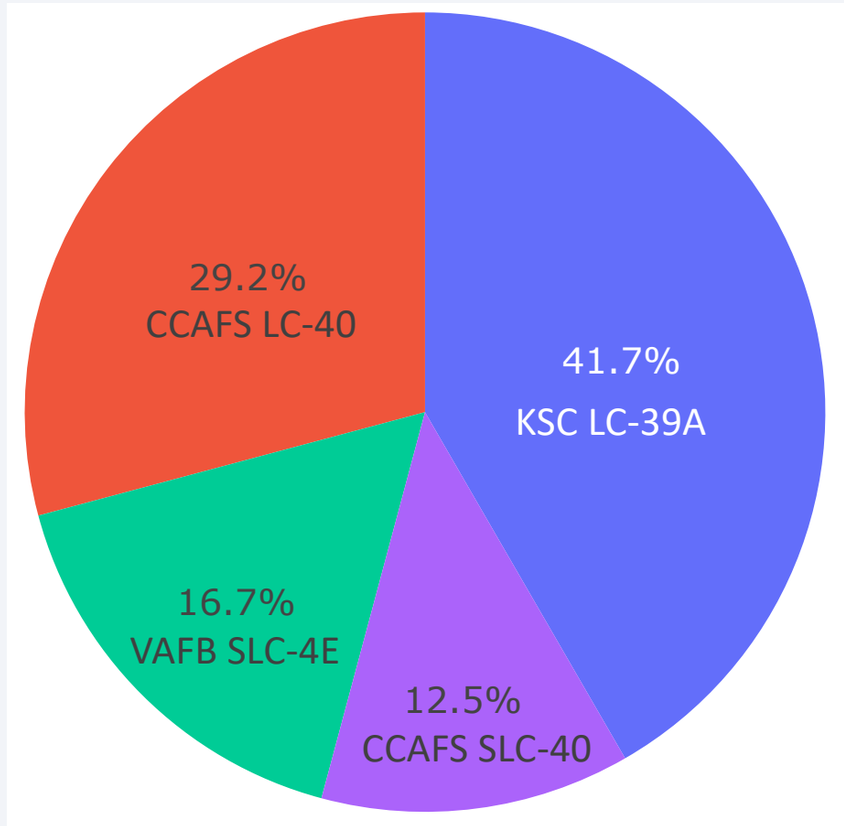
Build a Dashboard with Plotly Dash



Successful Launch Site

- Compared to the other sites, launch site **KSC LC-39A** hosted most successful launches:

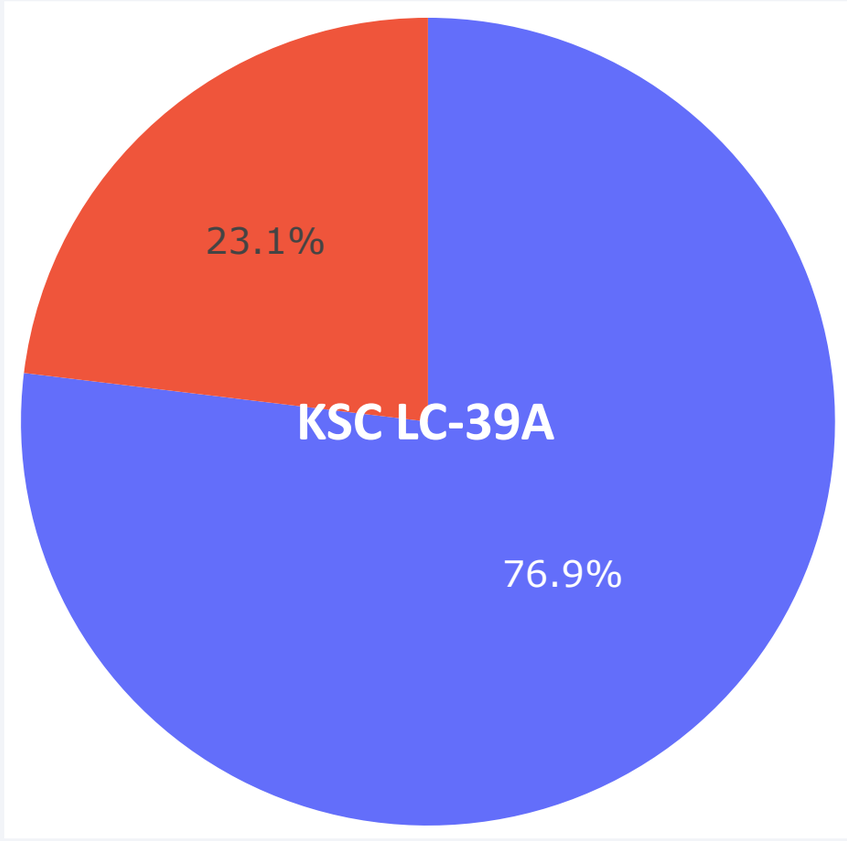
  41.7% of all successful launches happened here!



KSC LC-39A Launch Site Success

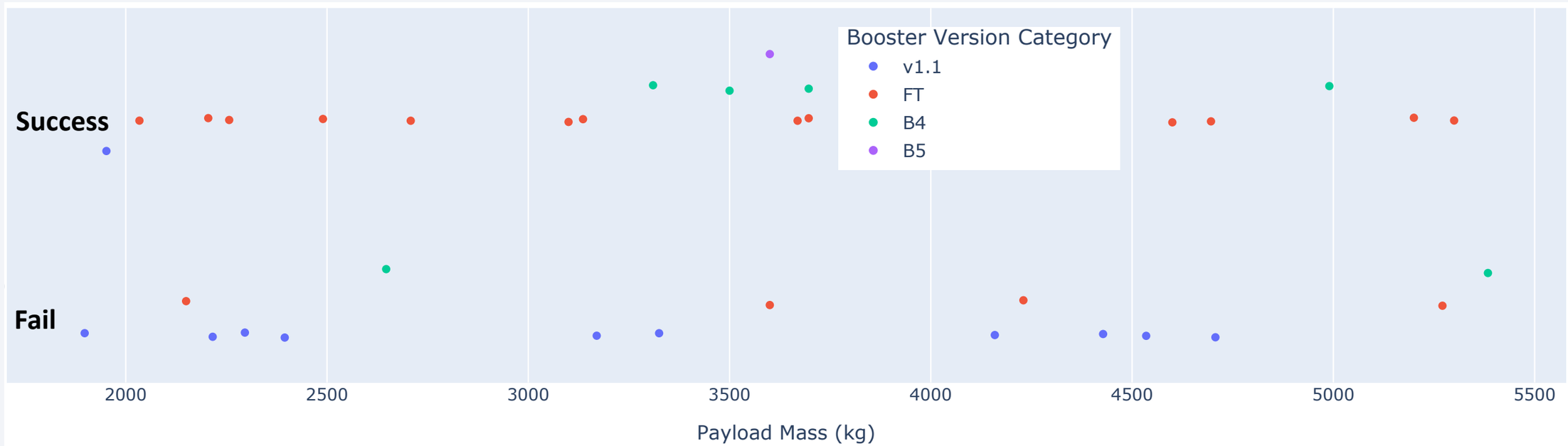
- **KSC LC-39A** had the highest proportion of successful launches:

  76.9% were successful



Success, Payload & Boosters

- Most successful launches are with payloads between **1.5** and **5.5 tons**
- The **FT** booster is most successful to return

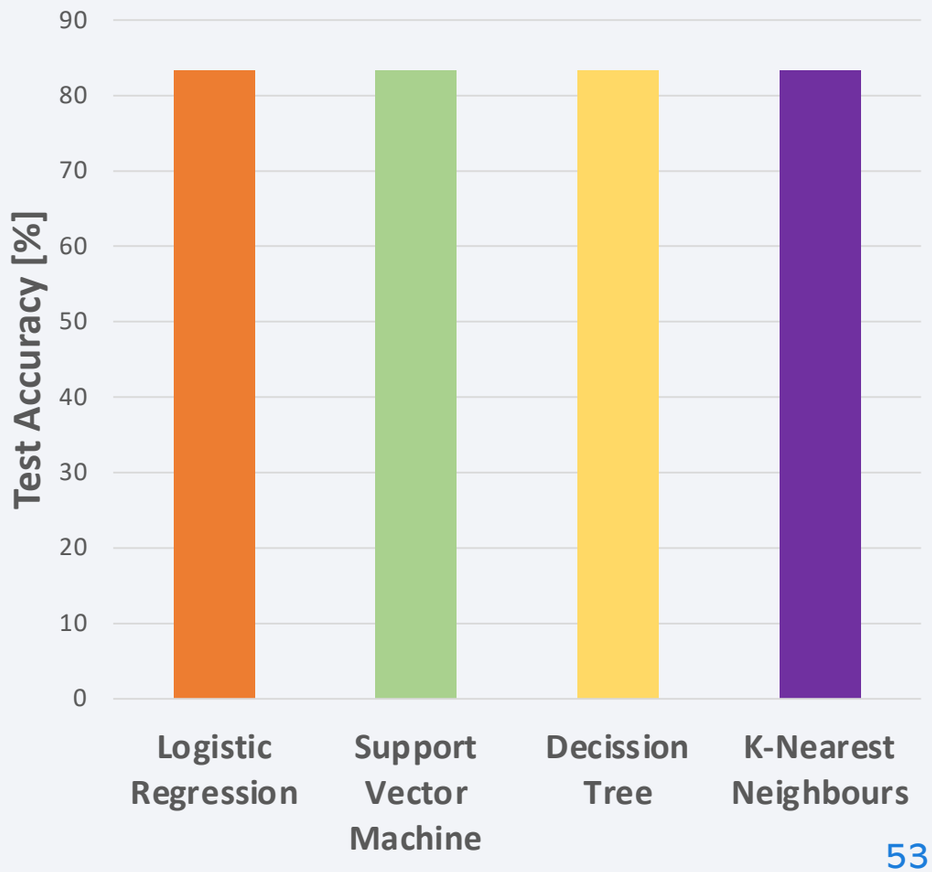


Section 6

Predictive Analysis (Classification)

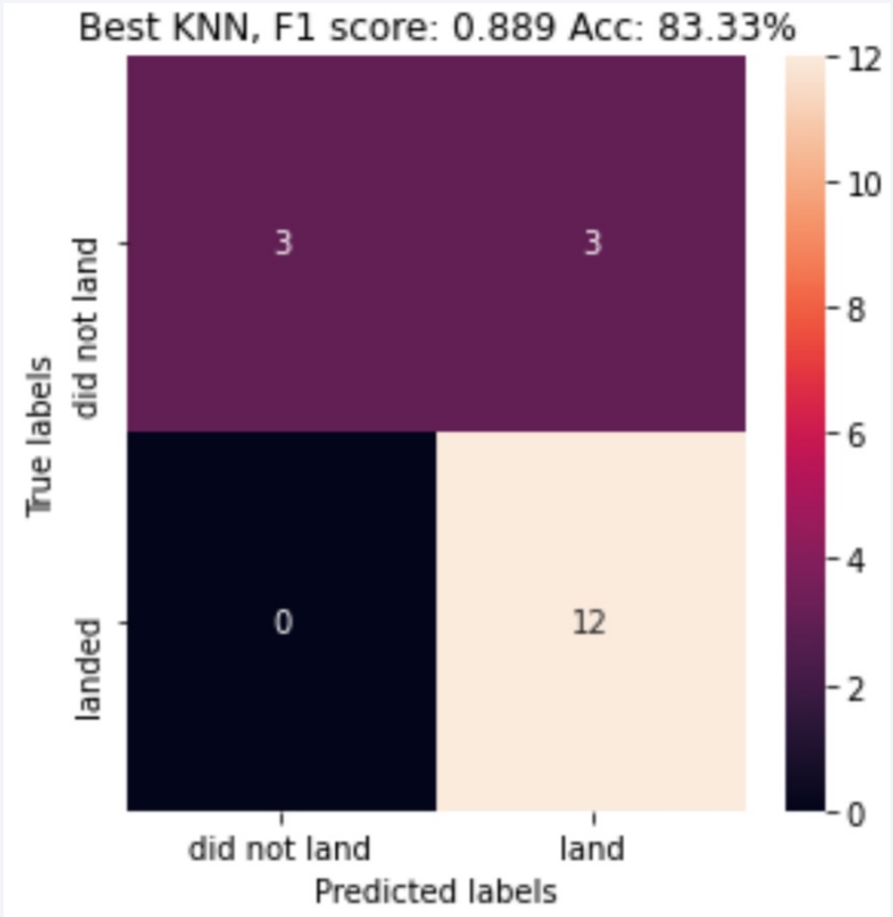
Classification Accuracy

- After grid search hyperparameter optimization all models performed similarly in predicting Falcon F9 first stage landings.
- The decision tree classifier did not always converge to the same optimal solution.



Confusion Matrix

- The confusion matrices of all best performing classifiers was similar:
  - True Positive: 3 out of 6
  - False Positive: 3 out of 6
  - True Negative: 0 out of 12
  - True Negative: 12 out of 12



Conclusions

- SpaceX has a very high mission success rate of 98%.
- 66.6% of Falcon F9 rocket first stages land successfully, most often on drone ships, and mostly the FT booster version.
- The Kennedy Space Center hosted the most successful launches for stage one returns. 76.9% returned.
- Cross-validation model training strategies could improve predictive performance and robustness.
- As the current model is likely to predict false successful landings, launch price estimates based on this model would be too low. The relatively high false positive rate must be accounted for.

Thank you!