IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jason Schmitz
11/13/2022

Outline

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

Executive Summary

- Summary of methodologies
  - SpaceX API and web scraping from SpaceX wikipedia page
  - EDA using sql, scatterplots, line graphs, and bar graphs
  - Interactive folium map and dashboard
  - Predictive analysis using LR, SVM, decision tree, and KNN
- Summary of all results
  - Graphs comparing payload, flight number, orbit type, launch site
  - Interactive folium map and dashboard
  - Results of predictive analysis accuracy scores and confusion matrices

Introduction

- Project background and context
  - The purpose of this report is to predict the likelihood that the first stage of the Falcon 9 rocket will land successfully in order to better identify the costs of the rockets. This information is important when comparing how SpaceX might stand out from potential competitors
- Problems you want to find answers
  - What features are most relevant in predicting successful landing outcomes
  - What machine learning model is best to predict future landing outcomes

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was obtained through the SpaceX API and web scraping launch data tables from the SpaceX Wikipedia page
- Perform data wrangling
    - Data was first filtered to include only Falcon 9 rocket data and stripped of irrelevant data leaving the data to be analyzed as Flight Number, Lauch Site, Payload Type, Payload Mass, Orbit Type, Customer, Launch Outcome, Booster Version, Landing Result, Date, Time, Grid Fins, Reused, Legs, Landing Pad, Block, and Reused Count
    - Payload mass data was missing from some of the launches so we used the mean mass as a substitute for these instances
    - The landing outcome was used to classify successful landings into a new column called "Class" where 0 is a failure and 1 is a success

Data Collection

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - KNN, Logisitic Regression, SVM and Decision Tree models were created. The data was standardized and split into test and train groups. The models were fit with the training data and the results recorded and accuracy score calculated

Data Collection – SpaceX API

1. Get response from API

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

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

2. decode and convert to dataframe

```
results = json.loads(response.text)
data=pd.json_normalize(results)
```

3. Get data

```
getBoosterVersion(data)
```

the list has now been updat

```
BoosterVersion[0:5]
```

['Falcon 1', 'Falcon 1',

we can apply the rest of the

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

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

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

4. Create 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': Longitud
'Latitude': Latitude}
```

5. convert to DataFrame

```
# Create a data from launch_dict
df=pd.DataFrame(launch_dict)
```

6. Filter Data for Falcon9

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df.loc[df['BoosterVersion']!='Falcon 1']
data_falcon9
```

7. Replace missing values

```
# Calculate the mean value of PayloadMass column
plm=data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass']=data_falcon9['PayloadMass'].replace(np.nan, plm)
data_falcon9
```

- API Calls

Data Collection - Scraping

1. HTML response and Beautiful soup

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

Create a `BeautifulSoup` object from the HTM

```
# Use BeautifulSoup() to create a Beautif
soup=BeautifulSoup(data,"html.parser")
```

2. Extract column names

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

Starting from the third table is our target tabl

```
# Let's print the third table and chec
first_launch_table = html_tables[2]
print(first_launch_table)
```

3. create dictionary

```
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']=[]
```

4. fill in data

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"w
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number co
        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 dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dic
            launch_dict["Flight No."].append(flight_number)
            datatimelist=date_time(row[0])
            #print(flight_number)


            # Date value
            # TODO: Append the date into launch_dict with ke
            date = datatimelist[0].strip(',')
            launch_dict["Date"].append(date)
            #print(date)

            # Time value
            # TODO: Append the time into launch_dict with ke
            time = datatimelist[1]
            launch_dict["Time"].append(time)
            #print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_dict["Version Booster"].append(bv)

            # Launch Site
            # TODO: Append the launch_site into launch_dict
            launch_site = row[2].a.string
            launch_dict["Launch site"].append(launch_site)
            #print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with
            payload = row[3].a.string
            launch_dict["Payload"].append(payload)
            #print(payload)
```

5. convert to dataframe

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

6. export to CSV

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

[web scraping](#)

Data Wrangling

1. Calculate # of launches

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

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

2. Calculate orbits

```
# Apply value_counts on Orbit colum
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3. Calculate outcomes

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

`True Ocean` means the mission outcome was successfully landed to
landed to a specific region of the ocean. `True RTLS` means the miss
unsuccessfully landed to a ground pad. `True ASDS` means the missic
unsuccessfully landed to a drone ship. `None ASDS` and `None None`

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

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

We create a set of outcomes where the second stage did not land suc

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

4. One hot encoding

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

```
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1,
0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

5. export to CSV

```
df.to_csv("dataset_part_2.csv", index=False)
```

Data Wrangling

EDA with Data Visualization

- Scatter plot comparing payload mass to flight number showing success rate increases as flight number increases and payload mass decreases
- Scatter plot comparing launch site to flight number showing success rate increases as flight number increases but launch site doesn't appear to be a factor
- Scatter plot comparing payload to launch site shows VAFB launch site had no payloads over 10,000kg
- Bar chart showing success rate per orbit type showing ES-L1, GEO, HEO, SSO, and VLEO have high success rates
- Scatter plot comparing orbit type by flight number showing success rate is related to flight number for LEO orbits but not for GTO orbits
- Scatter plot comparing orbit type by payload mass showing success rate for heavy payloads are higher for Polar, LEO, and ISS orbits
- Line graph showing success rate by year showing a general increasing success rate trend

[EDA with Data Visualization]()

EDA with SQL

- Found names of launch sites
- Found launch sites beginning with "CCA"
- Found total payload mass for boosters launched by NASA (CRS)
- Found average payload mass for F9 v1.1 booster
- Found date of first successful landing in ground pad
- Found names of boosters with success in drone ship and payload mass between 4,000 and 6000 kg
- Found total number of success and failure outcomes
- Found boosters that carried the max payload mass
- Found records for months in the year 2015
- Found the number of successful landings between 6/4/2010 and 3/20/2017 in descending order

[EDA with SQL]

Build an Interactive Map with Folium

- Added circles for each launch site to display the name of the site
- Added cluster at each site to show the successes and failures
- Added lines with distance markers to show the distance from the launch site to nearest coastline, railroad, highway, and city

[Folium Map]

Build a Dashboard with Plotly Dash

- Dropdown was added to select data for each launch site with and option to select all sites as well
- Pie Chart was added to show success rate per the launch site selected in the dropdown menu
- Range Slider was added to show data for the selected launch site by the payload mass size selected in the range slider
- Scatter plot was added that displayed the pay load mass data by launch site per the selections from the dropdown and range slider

Dashboard with Plotly Dash

Predictive Analysis (Classification)

1. Define Y (success rate)

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

2. Standardize X

```python
transform = preprocessing.StandardScaler()
X=transform.fit_transform(X)
```

3. Split into train and test

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

4. Create and test LR

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

5. Create and test SVM

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

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

6. Create and test D Tree

```python
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()
```

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

7. Create and test KNN

```python
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()
```

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

[Predictive Analysis]()

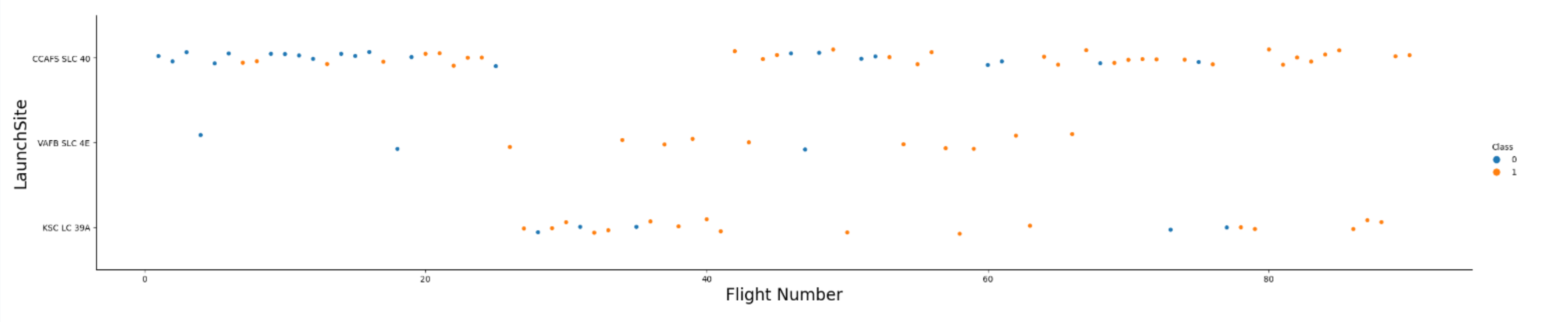
Results

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

Section 2

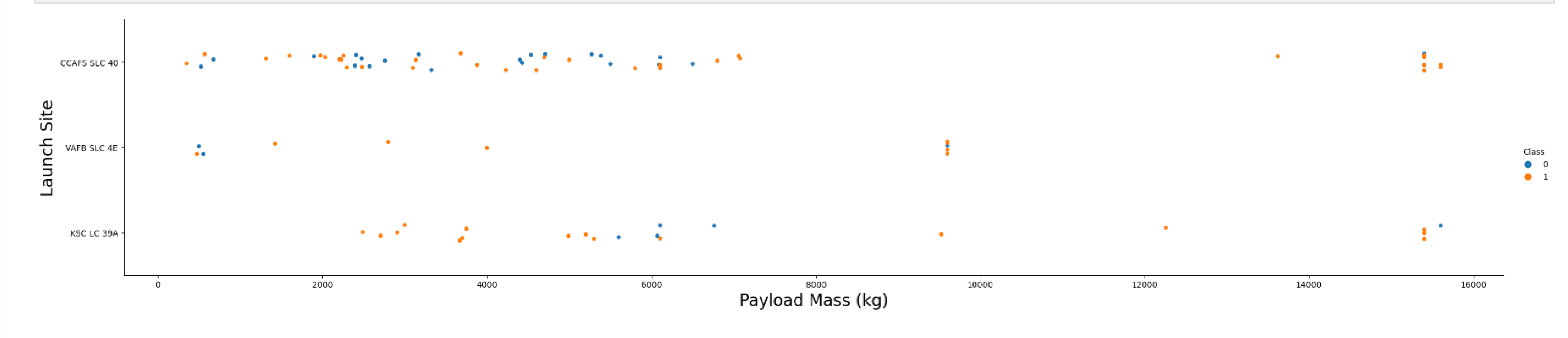
Insights drawn from EDA

Flight Number vs. Launch Site



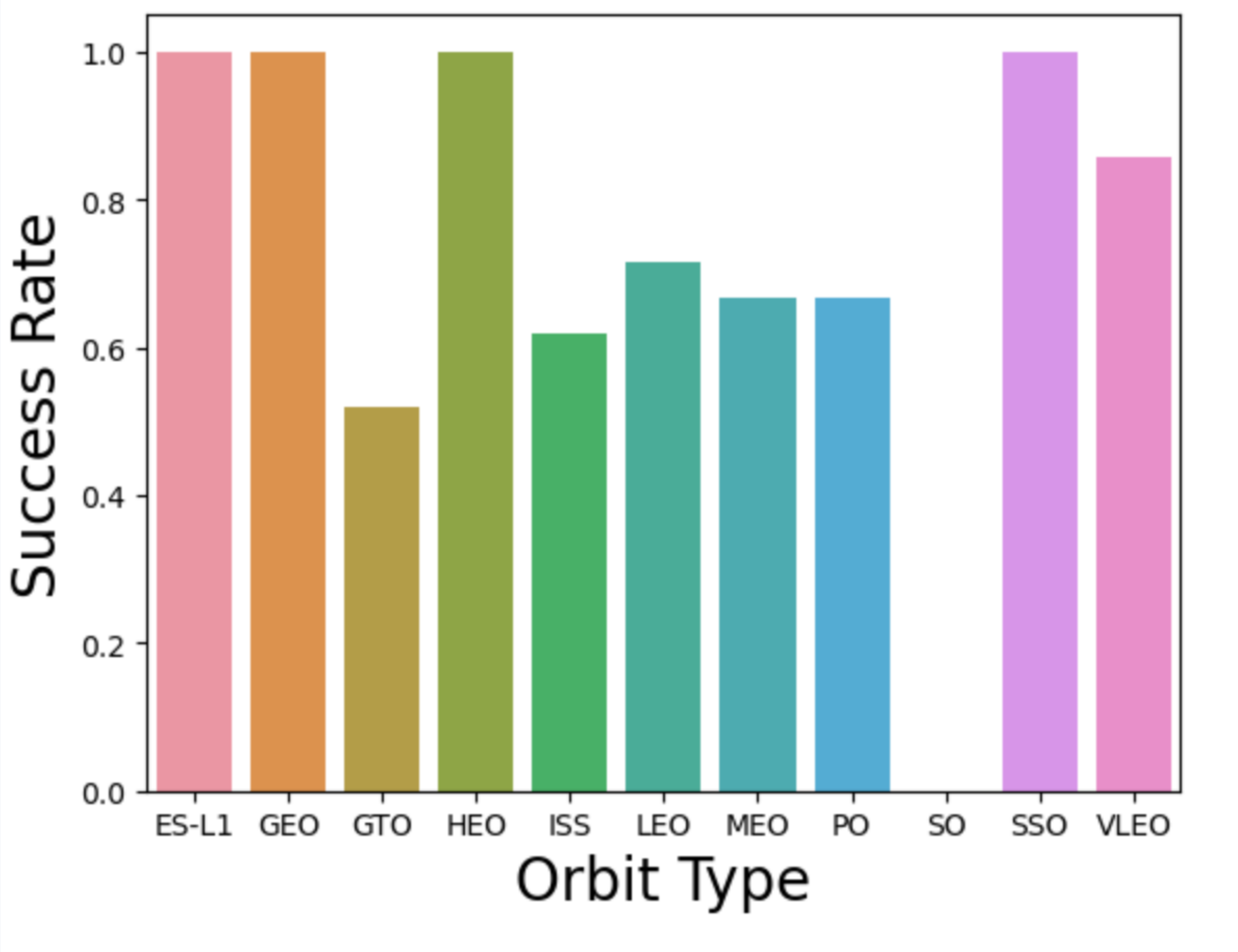
- Success rate increases as the number of flights increases but launch site does not appear to correlate with success or failure

Payload vs. Launch Site



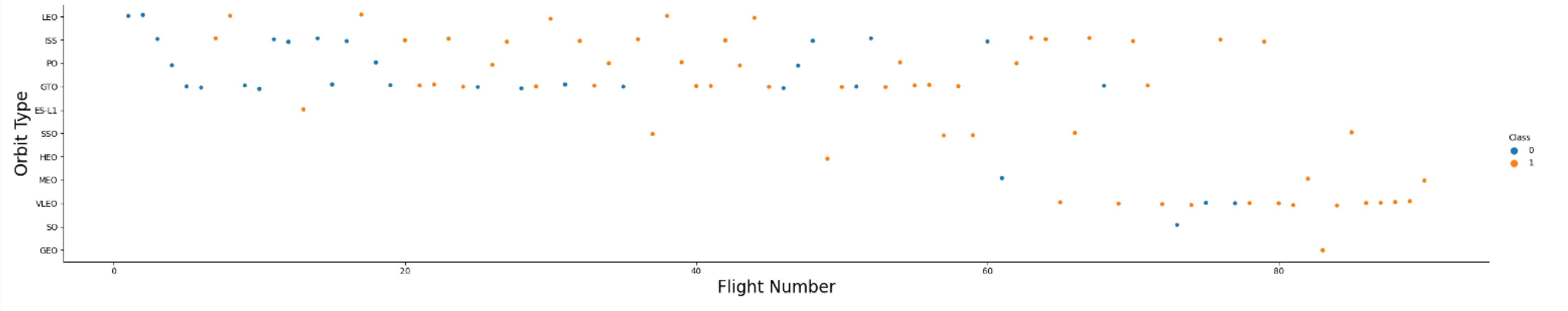
- VAFB SLC 4E has no payloads over 10,000kg

Success Rate vs. Orbit Type



- ES-L1, GEO, HEO and SSO all have 100% success rate with VLEO also being quite high
- SO has no successful landings
- The rest of the orbit types are between 50% and 70% success rate

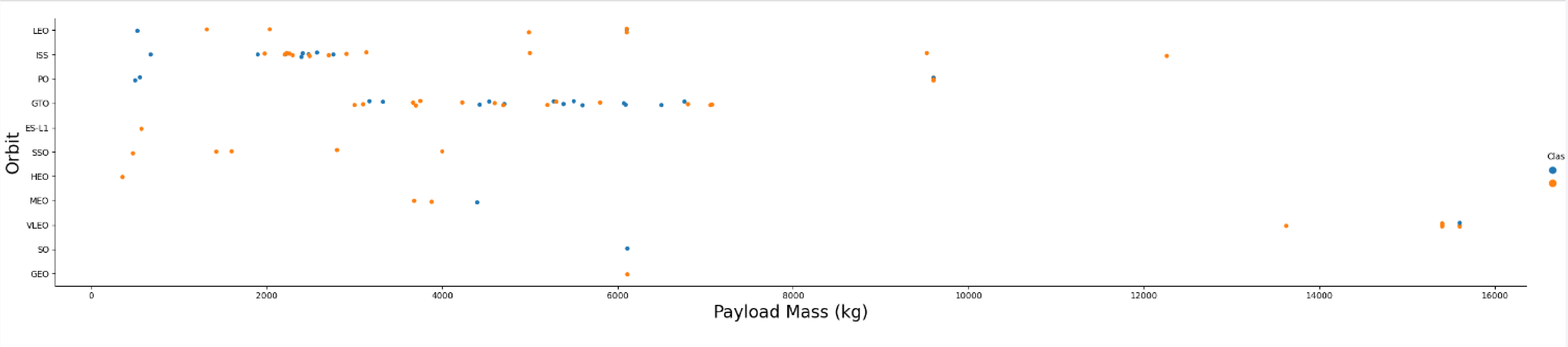
Flight Number vs. Orbit Type



- LEO orbit success rate is related to flight number

- GTO orbit's success rate is not related to flight number

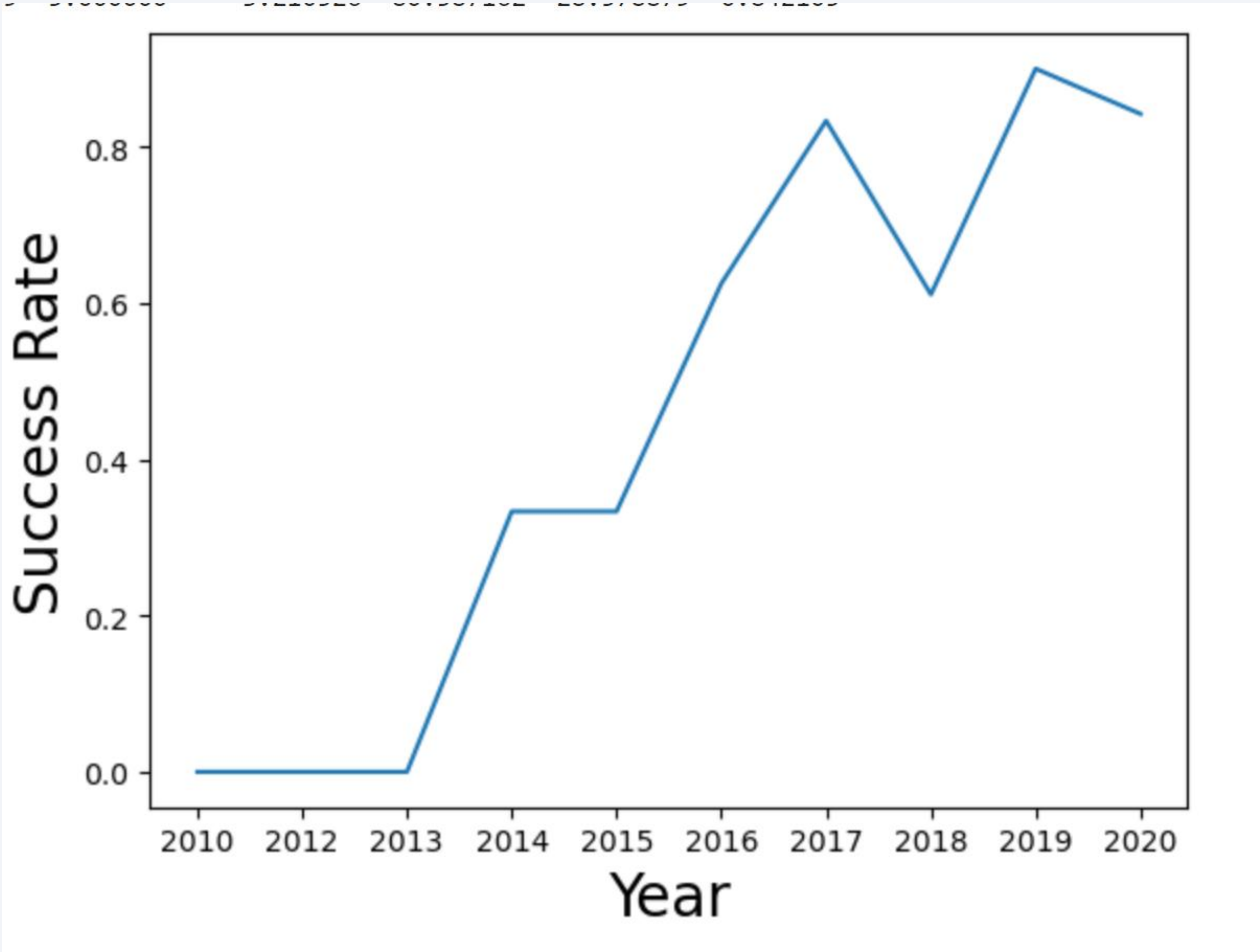
Payload vs. Orbit Type



- Heavy payloads have a higher success rate with Polar, LEO, and ISS orbits

- GTO orbit success rates are not related to payload mass

Launch Success Yearly Trend



- Success rate trend is increasing year over year with a slight dip in 2018 and 2020

All Launch Site Names

```
%sql select distinct Launch_Site from spacextbl
```

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

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

- Launch site names

Launch Site Names Begin with 'CCA'

```
%sql select * from spacextbl where Launch_Site like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landi _Outcor
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failu (parachu
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	Failu (parachu
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	I atten
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	I atten
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	I atten

Launch sites beginning with "CCA"

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer like '%NASA (CRS)%'
```

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

sum(PAYLOAD_MASS__KG_)
48213

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version like '%F9 v1.1%'
```

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

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

```
%sql select min(date) from spacextbl where Landing_Outcome like '%Success (ground pad)%'
```

MIN("DATE")
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select distinct Booster_Version from spacextbl where Landing_Outcome like '%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 Number of Successful and Failure Mission Outcomes

```
%sql select Mission_Outcome, count(Mission_Outcome) as Total from spacextbl group by Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

```
%sql select Booster_Version from spacextbl where Payload_Mass__Kg_ = (select max(Payload_Mass__Kg_) from spacextbl)
```

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

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

ss

2015 Launch Records

```
%sql select * from spacextbl where substr(Date,7,4)='2015' and Landing_Outcome like '%Failure (drone ship)%' order by substr(Date,4,2)
```

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

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

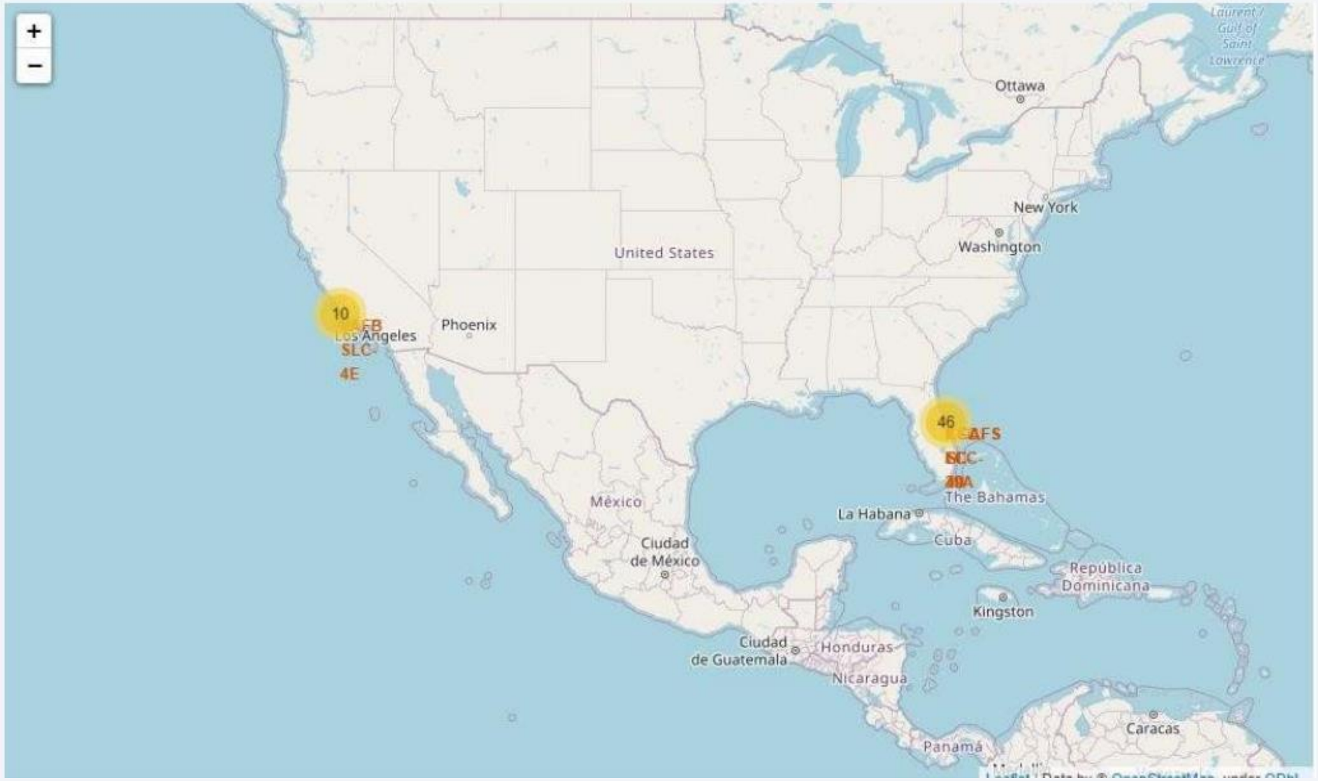
```
%sql select Landing_Outcome, count(*) as Number_of_Successes from spacextbl where date between 04-06-2010 and 20-03-2017 ORDER BY Number_of_Successes
```

Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

Launch Sites Proximities Analysis

Launch Site Locations

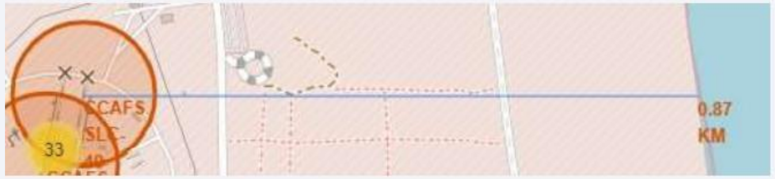


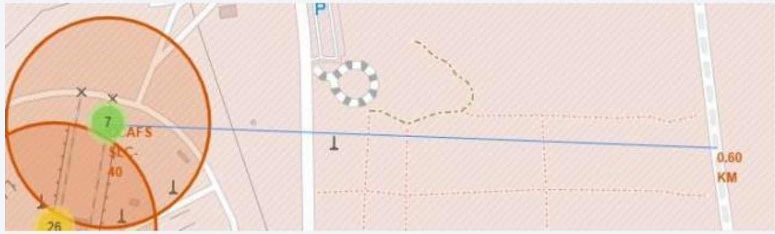
- The circles indicate each launch site with the number of launches and name of the site visible

Launch Site Successes and Failures

- This is the cluster marker for each launch site with green representing success and red representing failure

Distances from Launch Site



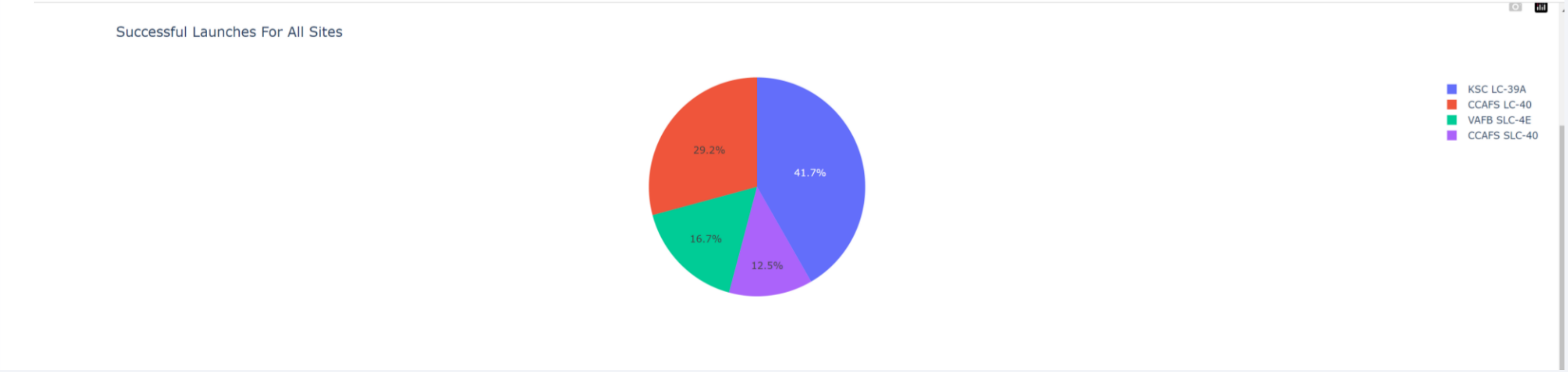


- Top left is distance to nearest coastline
- Middle left is distance to nearest highway
- Bottom left is distance to nearest city
- Right is the distance to the nearest railway

Section 4

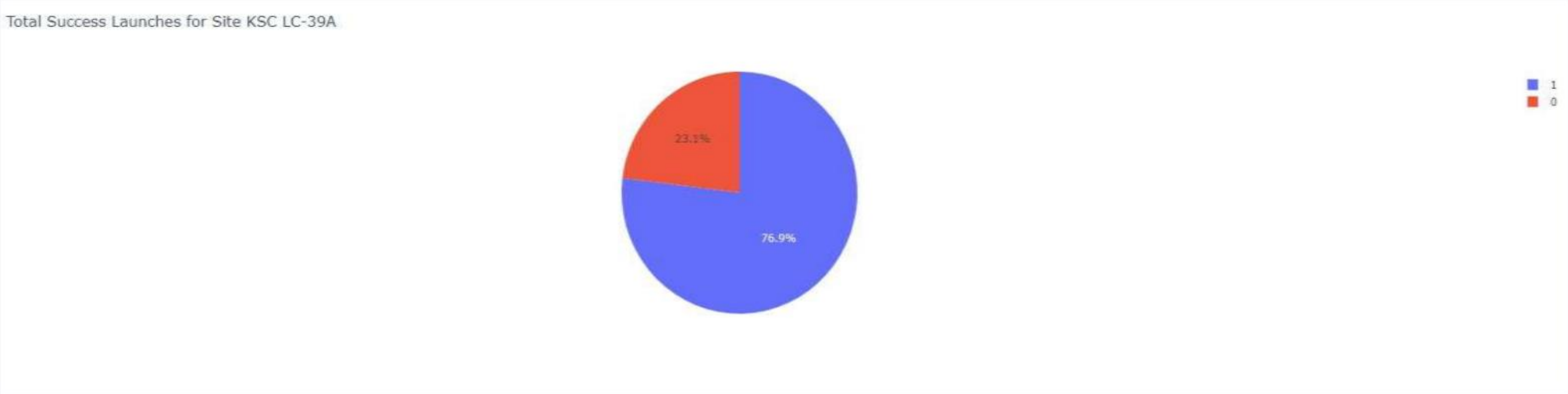
Build a Dashboard with Plotly Dash

Success rate for all sites



- KSC LC-39A has the highest success rate
- CCAFS SLC-40 has the lowest success rate

KSC LC-39A success rate



- Success rate for KSC LC-39A is 76.9%

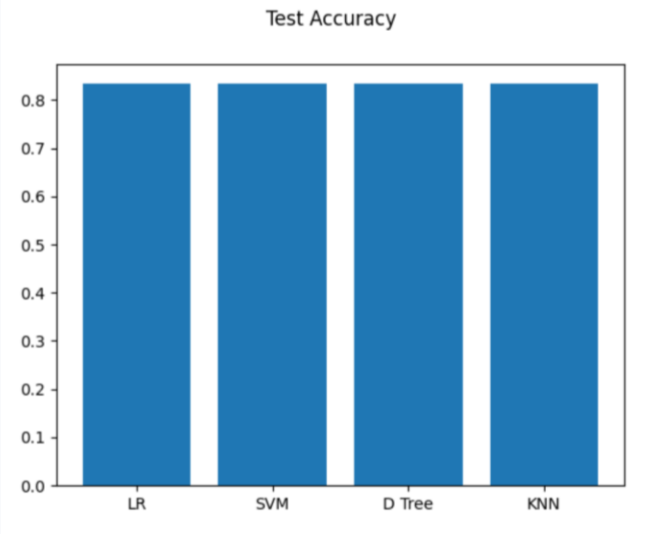
Payload mass effect for all sites

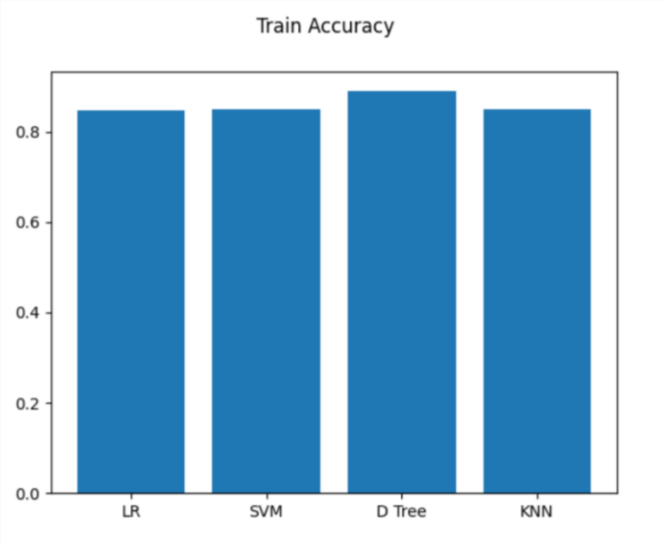
- Top is showing successes with payload from 0-5000kg
- Bottom is showing successess with payload from 5000-10000kg

Section 5

Predictive Analysis (Classification)

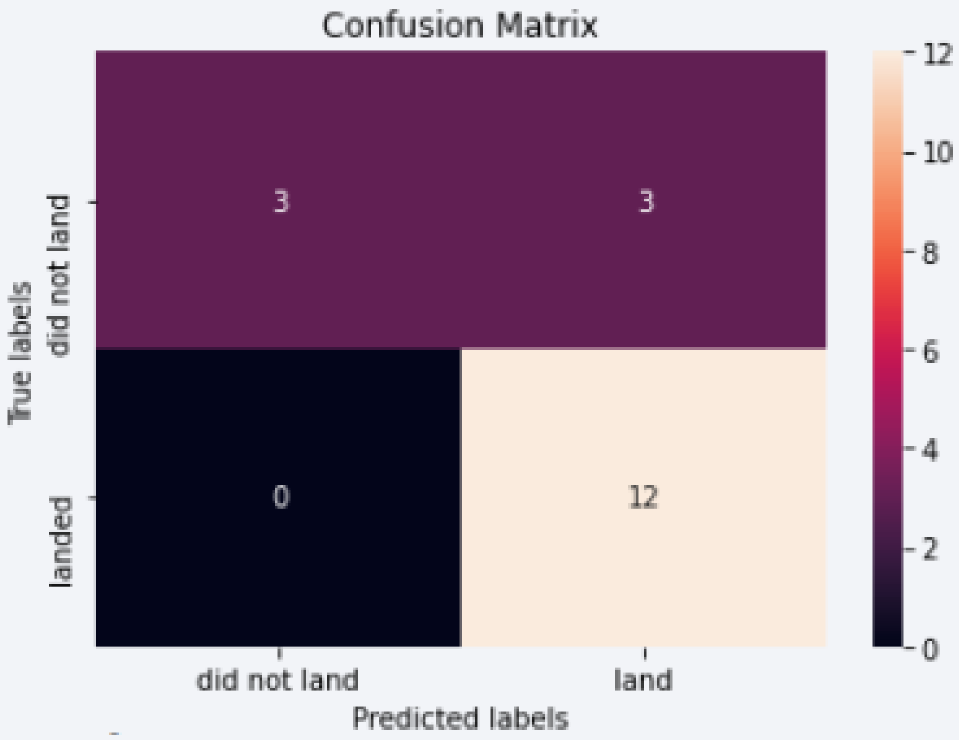
Classification Accuracy





- The Decision tree model was slightly better than the others on the training data at 88.9% and so the Decision Tree model should be used

Confusion Matrix



- The confusion matrix for all 4 models were exactly the same as well showing that the model can distinguish between classes but can give us false positives

Conclusions

- In order to maximize success rate of 1st stage rocket landings, payload mass, orbit type, launch site, and booster version should be taken into consideration
- Decision Tree algorithm should be used to determine success rate for future launches
- Success can be maximized by keeping the payload mass low and the orbit low
- Additional data should be obtained on future launches to see if the noise of most of the early launches being failures

Appendix

- link to all assignments
- SpaceX Wikipedia

Thank you!