ANALYSIS OF SPACEX LAUNCH DATA

Predicting when Falcon 9 landings are successful... or not

Executive Summary

- Introduction
- Methodology
  - Data Collection
  - Exploratory Analysis and Visualization
  - Predictive Analysis
- Results
  - Exploratory Visualization and SQL results
  - Interactive Map with Folium
  - Predictive Model results and scores/metrics
- Conclusion

Introduction

Background

In the business of launching rockets into space, one of the key factors is, of course, the cost.

SpaceX has been able to offer launches at a lower cost in part because they have been developing rockets with which they are able to land the first stage booster back to Earth for it to be reused, instead of discarding it.

While this is not always successful, it has huge cost saving potential. (To the tune of tens of millions of dollars.)

In this project we aim to collect and use data from SpaceX Falcon 9 launches to predict whether the first stage will successfully land for a given launch.

Introduction

Goals

- Gather available data for Falcon 9 launches
- Determine what factors influence the success or failure of landing the rocket
- Test and compare the performance of multiple methods/models for predicting success/failure of the landing based on the selected features
- See which features give the best chance of success

Methodology

Methodology

Data collection

- Data was collected into Pandas DataFrames with Python using the “requests” package to access SpaceX API and web scraping from Wikipedia using requests and “BeautifulSoup” packages

Methodology

Data collection

- Data was collected into Pandas DataFrames with Python using the “requests” package to access SpaceX API and web scraping from Wikipedia using requests and “BeautifulSoup” packages
- API Example Code:

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        if load:
         response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
         PayloadMass.append(response['mass_kg'])
         Orbit.append(response['orbit'])
```

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
           response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
           Longitude.append(response['longitude'])
           Latitude.append(response['latitude'])
           LaunchSite.append(response['name'])
```

Methodology

Data collection

- Web Scraping Example Code:

```
response = requests.get(static_url)
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text)
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

```
def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate(table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out

def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass
```

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

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        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_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])

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

Methodology

Data Wrangling/Cleaning

- With the data in a Pandas data frame, it was then preprocessed to select relevant features, remove/replace null values where applicable
- Keep only Falcon 9 launches, deleting others
- Created a new column called "Class" to be the target variable for prediction (1 for successful, 0 for failure)
- Used One-Hot encoding to transform categorical features into numerical format with Pandas get_dummies() method

Methodology

Exploratory Analysis

Used seaborn and pyplot packages to visualize how some of the variables relate to one another as well as the landing success rate

Methodology

Exploratory Analysis

Used seaborn and pyplot packages to visualize how some of the variables relate to one another as well as the landing success rate

Also used SQL queries to examine certain aspects of the data, such as:

- Success and failure of landings based on the type of landing pad. Some are done on land, some are done in the ocean.
- Average mass of payloads carried by Booster Versions
- Check the unique launch sites

Created an interactive map using the Folium package, showing launch sites and adding markers showing successful and failed landings.

```python
# Initialize the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
for index, row in launch_sites_df.iterrows():
    coordinates = [row['Lat'], row['Long']]
    # Create a orange circle at NASA Johnson Space Center's coordinate with a popup label showing its name
    circle = folium.Circle(coordinates, radius=1000, color='#d35400', fill=True).add_child(folium.Popup(row['Launch Site'])).add_to(site_map)
    # Create a orange circle at NASA Johnson Space Center's coordinate with a icon showing its name
    marker = folium.map.Marker(
        coordinates,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % row['Launch Site'],
            )
        ).add_to(site_map)
```

Methodology

Exploratory Analysis

Used seaborn and pyplot packages to visualize how some of the variables relate to one another as well as the landing success rate

Also used SQL queries to examine certain aspects of the data, such as:

- Success and failure of landings based on the type of landing pad. Some are done on land, some are done in the ocean.
- Average mass of payloads carried by Booster Versions
- Check the unique launch sites

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
average_by_year = df.groupby("Year")['Class'].mean()

plt.plot(average_by_year.index, average_by_year.values, 'co-')
plt.xlabel("Year")
plt.ylabel("Landing Success Rate")
plt.title("Landing Success Rate vs Year")
plt.show()
```

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 2)
plt.xlabel("Flight Number",fontsize=15)
plt.ylabel("Payload Mass (kg)",fontsize=15)
plt.title("Payload Mass vs Flight Number - Successful(Orange) and Unsuccessful(Blue) Landings")
plt.xticks(ticks=np.arange(-1,90, step=5))
plt.show()
```

```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 2)
plt.xlabel("Payload Mass (kg)",fontsize=15)
plt.ylabel("Launch Site",fontsize=15)
plt.title("Launch Site vs Payload Mass - Successful(Orange) and Unsuccessful(Blue) Landings")
plt.show()
```

```
# Success Rate by Orbit Type
orbit_success = pd.DataFrame(df.groupby('Orbit')['Class'].mean())
sns.barplot(x="Orbit",y="Class",data=orbit_success,hue='Class', legend=False)
plt.title("Percentage of Successful Landings by Orbit Type")
```

```
# Success Rate by Launch Site
site_success = pd.DataFrame(df.groupby('LaunchSite')['Class'].mean())
sns.barplot(x="LaunchSite",y="Class",data=site_success,hue='Class', legend=False)
plt.title("Percentage of Successful Landings by Launch Site")
```

Methodology

Predictive Analysis

- Used scikit-learn package to split data into training and testing sets, normalize the features data, then use GridSearchCV() method with 4 different machine learning classifiers and check their accuracy

- Models used: Logistic Regression, Support Vector Classifier, Decision Tree Classifier, and K Nearest Neighbors

- Plotted confusion matrix with seaborn for each model's test data predictions

```
X = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-D
Y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
parameters ={"C":[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
lr=LogisticRegression()

logreg_cv = GridSearchCV(lr,parameters,cv=10)
logreg_cv.fit(X_train, Y_train)
print("tuned hpyerparameters :(best parameters) ", logreg_cv.best_params_)
```

```
print("test set accuracy :",logreg_cv.score(X_test, Y_test))
```

```
test set accuracy : 0.8333333333333334
```

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

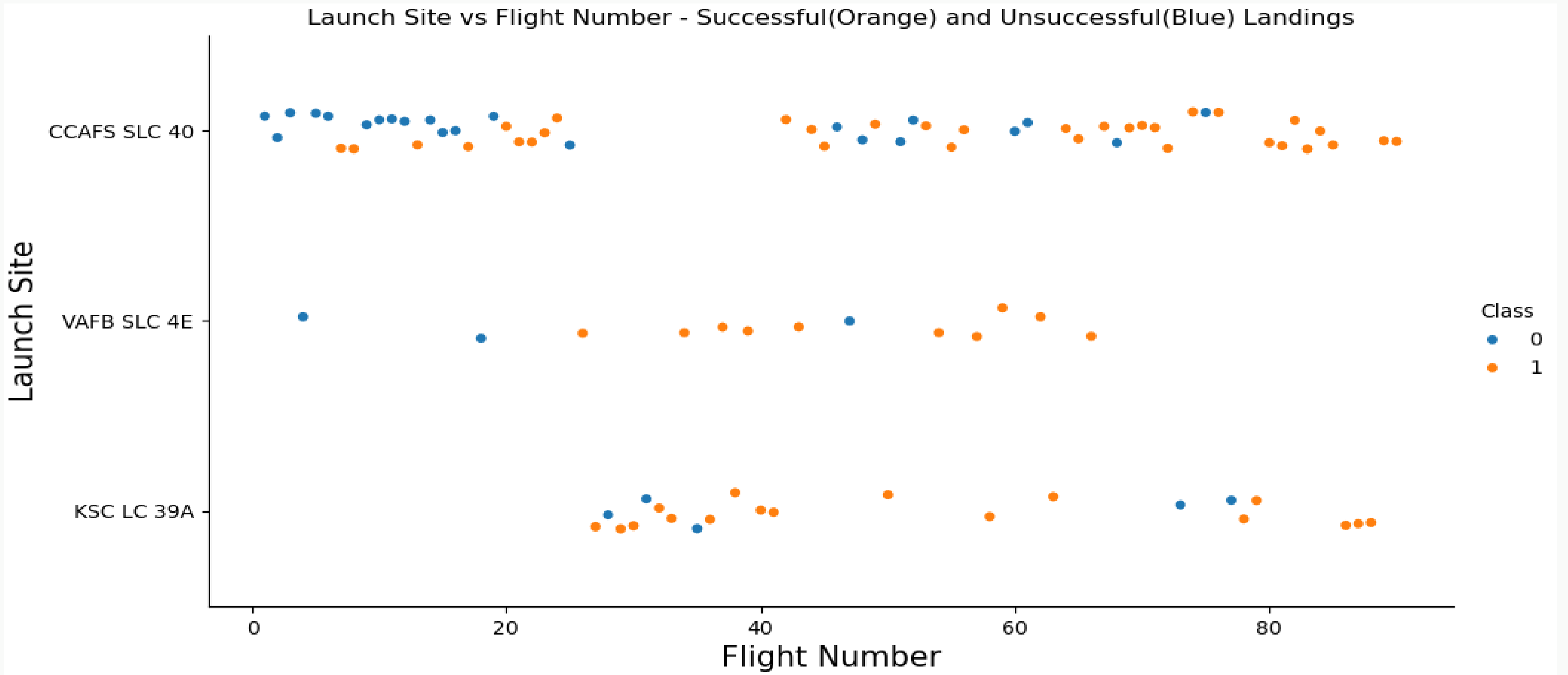
Results

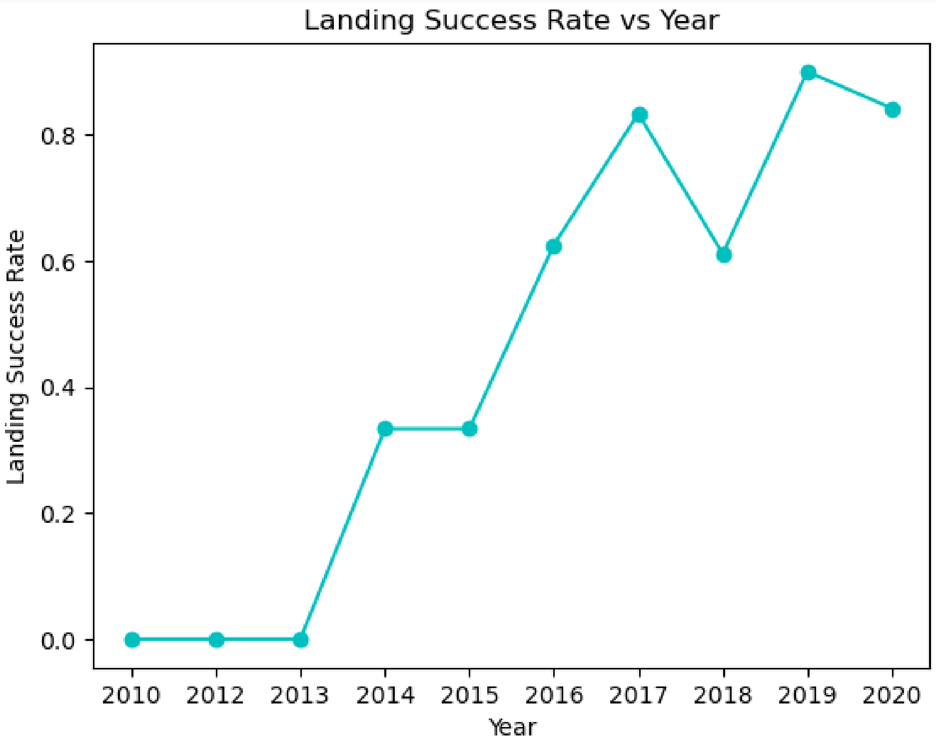
Results

Exploratory Analysis

From the first catplot we noticed that with higher flight numbers, the success rate was higher

A line graph showing the success rate over time shows us the same thing. SpaceX has gotten better at landing the booster as they've gotten more experience attempting it.



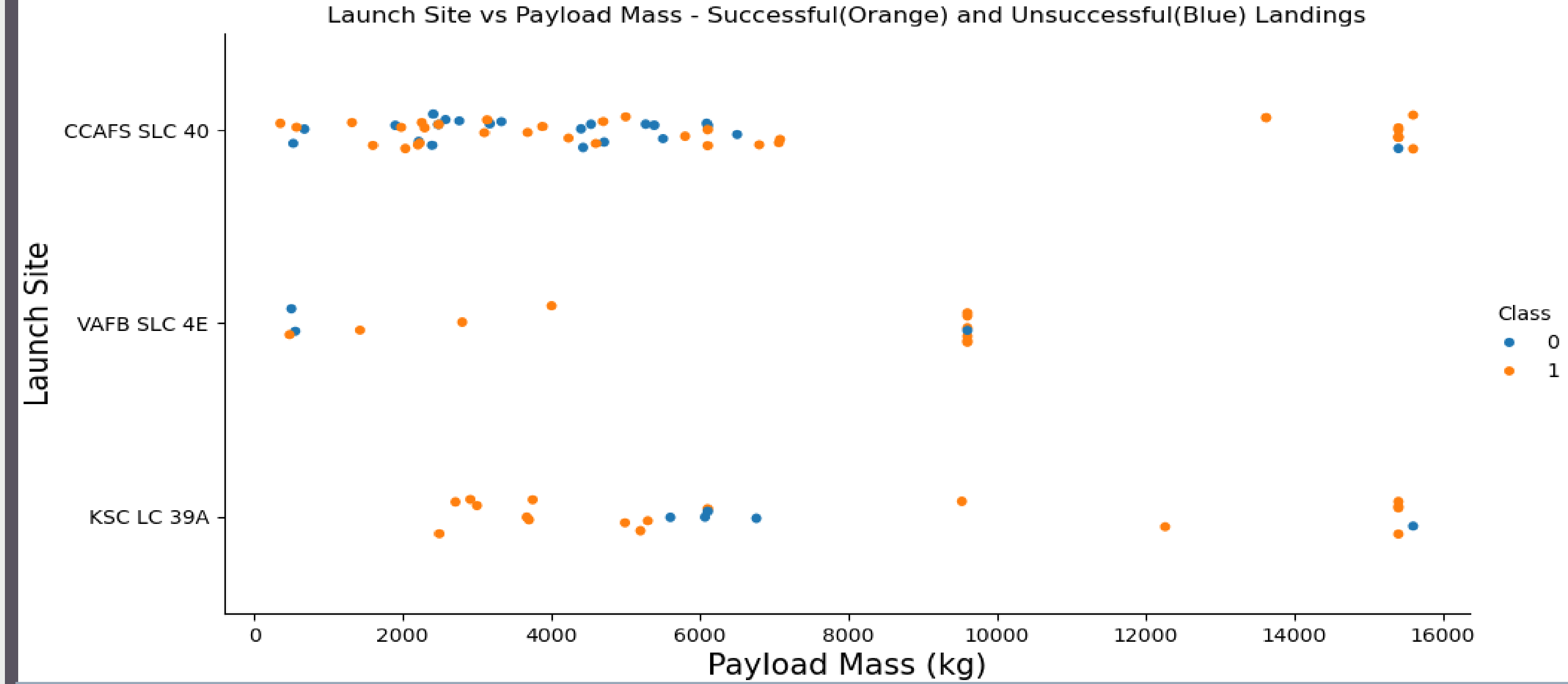


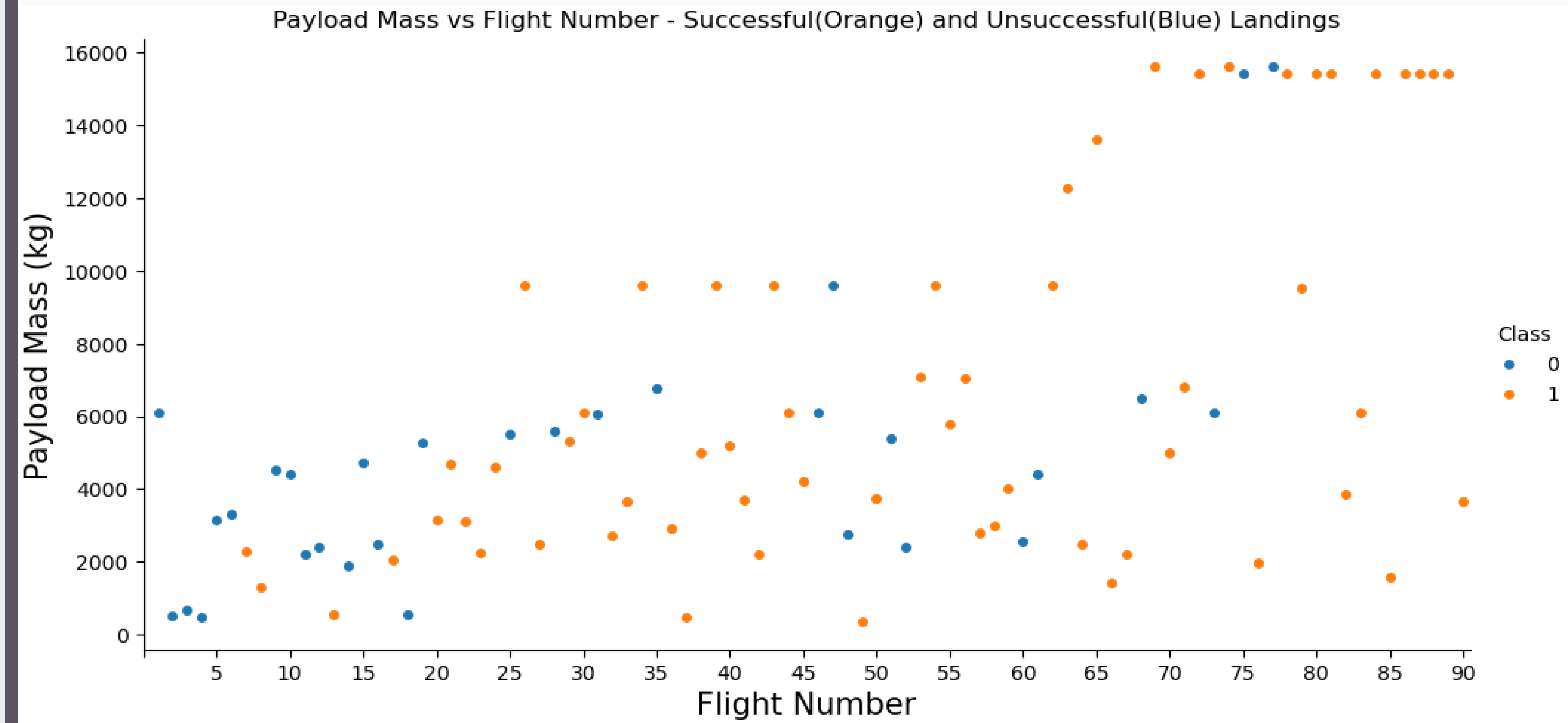
Results

Exploratory Analysis

The Vandenberg Space Launch Complex in California has not had any pay loads at 10,000 kg or higher, while the launch sites in Florida have.

SpaceX has gradually increased the maximum pay load masses carried by Falcon 9 over time and the larger pay loads do not seem to negatively affect the booster landing success rate. The success rate still increases.



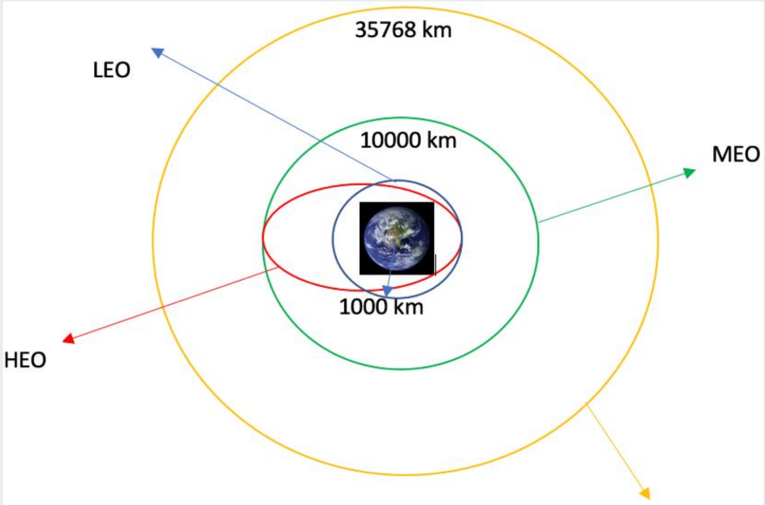


Results

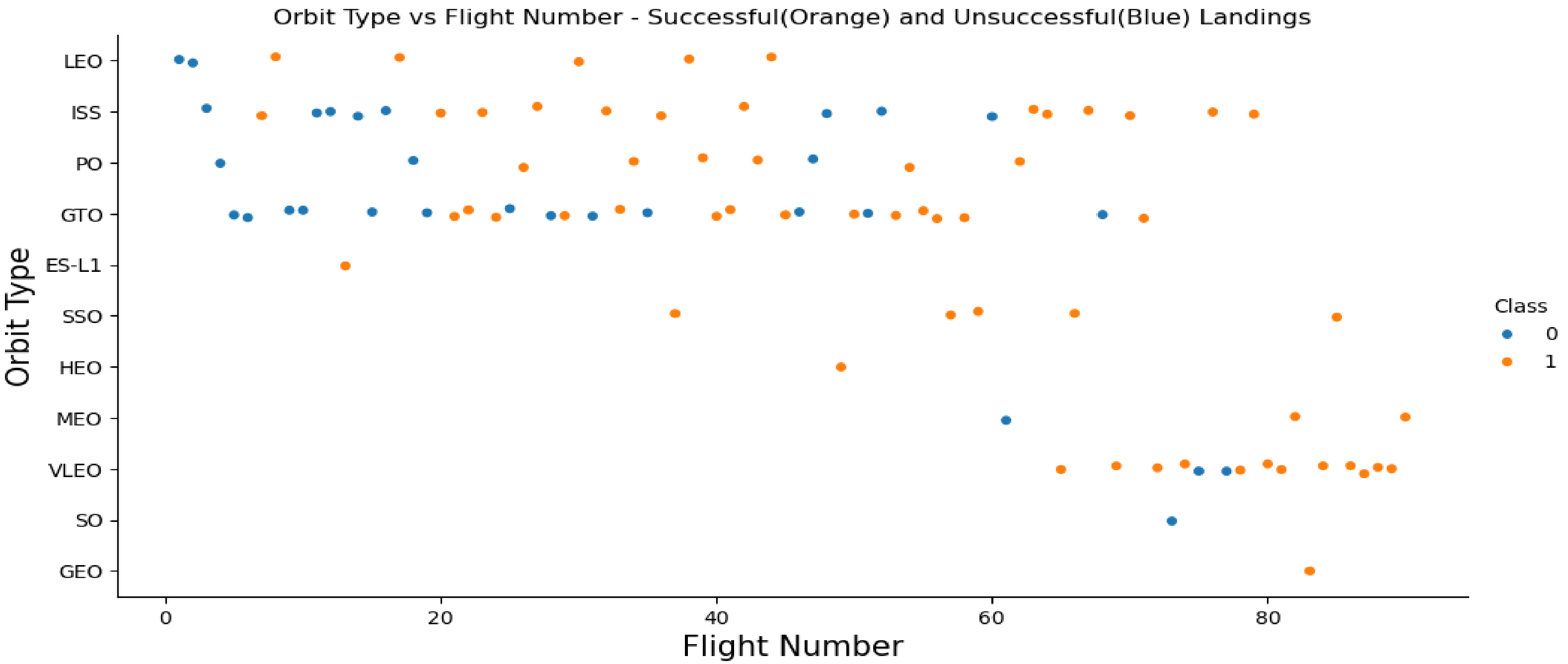
Exploratory Analysis

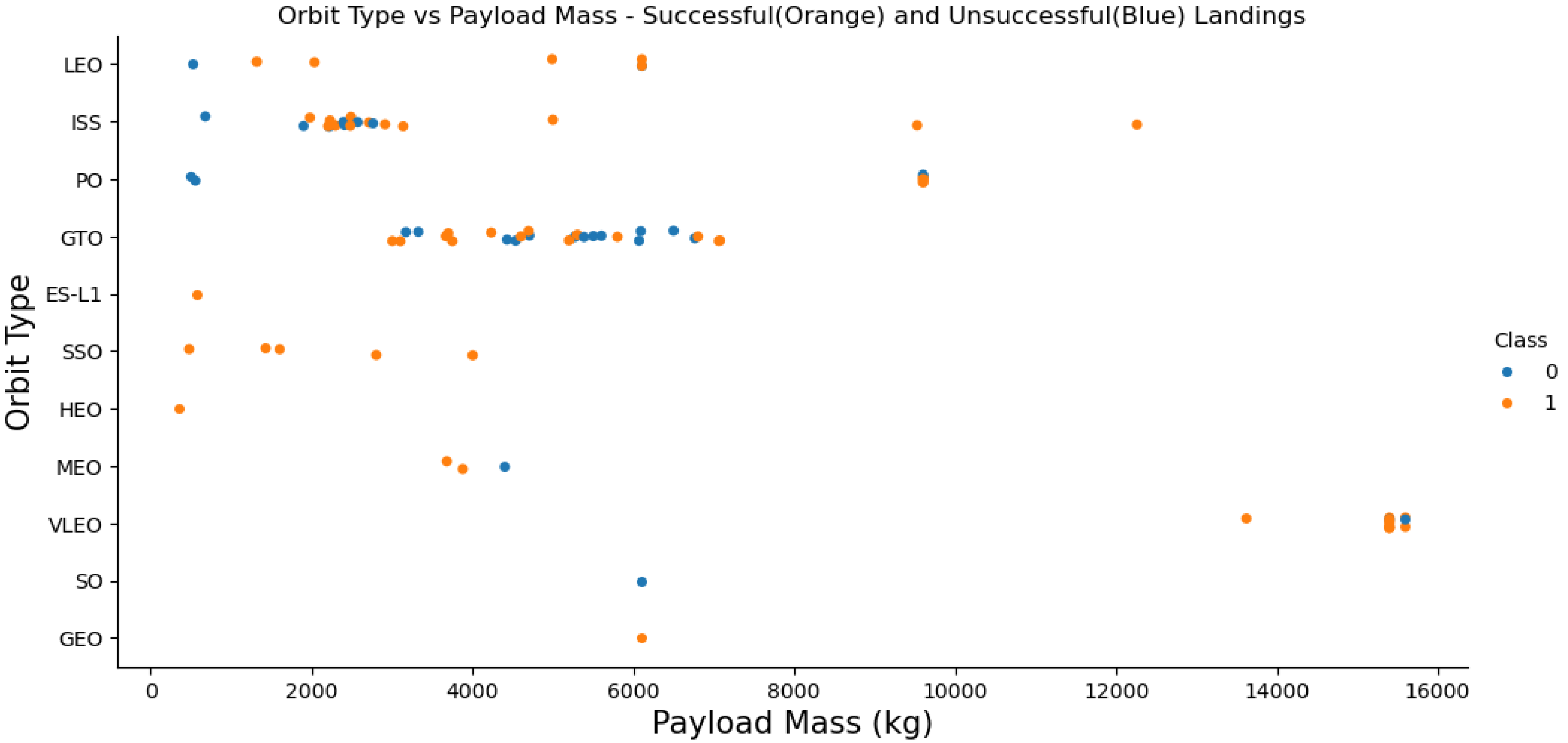
Launches have expanded into a larger variety of Orbit Types over time

ISS and PO (Polar Orbit) have some launches with higher than average pay load mass, while VLEO (Very Low Earth Orbit) has several launches with significantly higher mass than all other types
ISS and GTO (Geosynchronous Transfer) launches fail to land more often than others



Orbit Type Examples

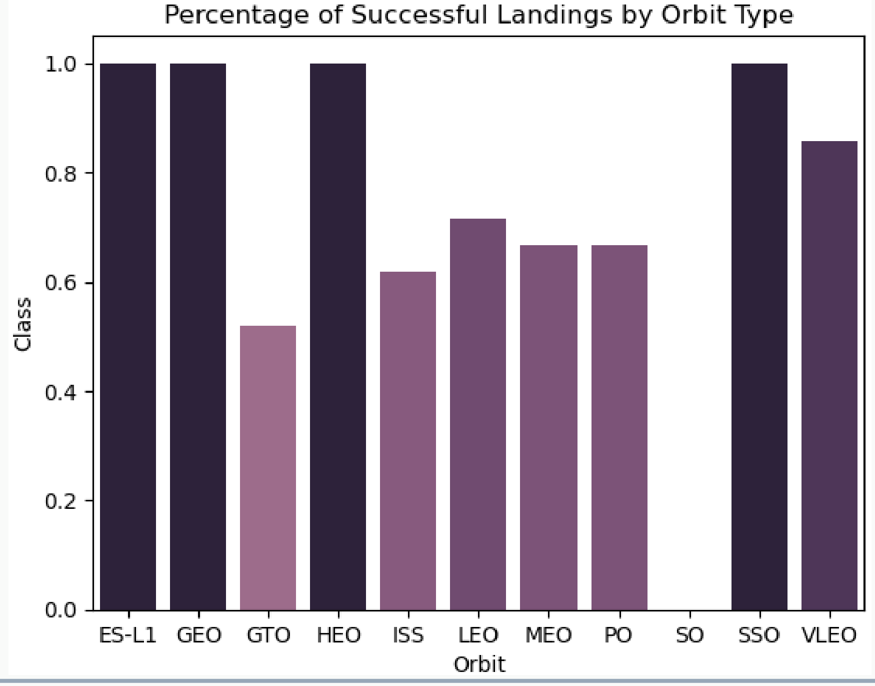


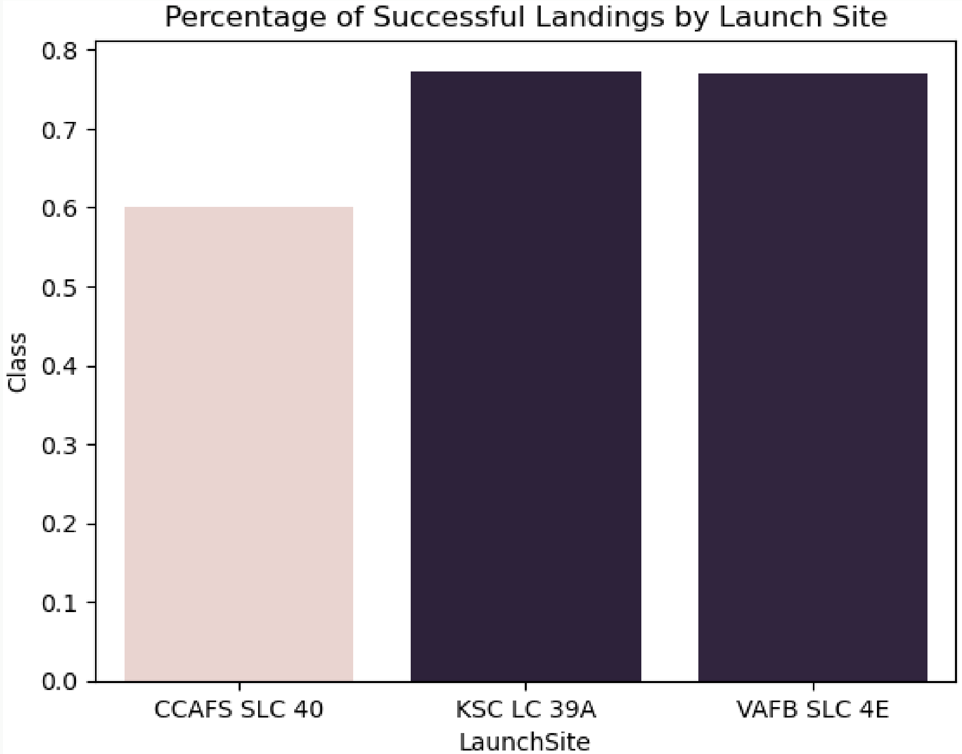


Results

Exploratory Analysis:

Bar Charts showing Landing Success Rates for different Orbit Types and Launch Sites





Orbit Type Examples

Results

Exploratory Analysis: SQL

Landing Outcomes by Type of Landing:

```
%%sql
select
    COUNT(*) as count,
    "Landing_Outcome"
from SPACEXTABLE
group by "Landing_Outcome"
order by count desc
```

count	Landing_Outcome
38	Success
21	No attempt
14	Success (drone ship)
9	Success (ground pad)
5	Failure (drone ship)
5	Controlled (ocean)
3	Failure
2	Uncontrolled (ocean)
2	Failure (parachute)
1	Precluded (drone ship)
1	No attempt

Average Pay Load Mass by Booster Version:

```
%%sql
select "Booster_Version", AVG("PAYLOAD_MASS__KG_") as "Booster_mean_payload_mass"
from SPACEXTABLE
group by "Booster_Version"
```

Booster_Version	Booster_mean_payload_mass
F9 B4 B1039.2	2647.0
F9 B4 B1040.2	5384.0
F9 B4 B1041.2	9600.0
F9 B4 B1043.2	6460.0
F9 B4 B1039.1	3310.0
F9 B4 B1040.1	4990.0
F9 B4 B1041.1	9600.0
F9 B4 B1042.1	3500.0
F9 B4 B1043.1	5000.0
F9 B4 B1044	6092.0
F9 B4 B1045.1	362.0
F9 B4 B1045.2	2697.0
F9 B5 B1046.1	3600.0
F9 B5 B1046.2	5800.0
F9 B5 B1046.3	4000.0
F9 B5 B1046.4	12050.0
F9 B5 B1047.2	5300.0
F9 B5 B1047.3	6500.0
F9 B5 B1048.2	3000.0
F9 B5 B1048.3	4850.0

(96 total versions
20 shown)

Boosters which have carried the Highest Pay Load (15,600 kg):

```
%%sql
select "Booster_Version"
from SPACEXTABLE
where
    "PAYLOAD_MASS__KG_" = (select MAX("PAYLOAD_MASS__KG_") from SPACEXTABLE)
```

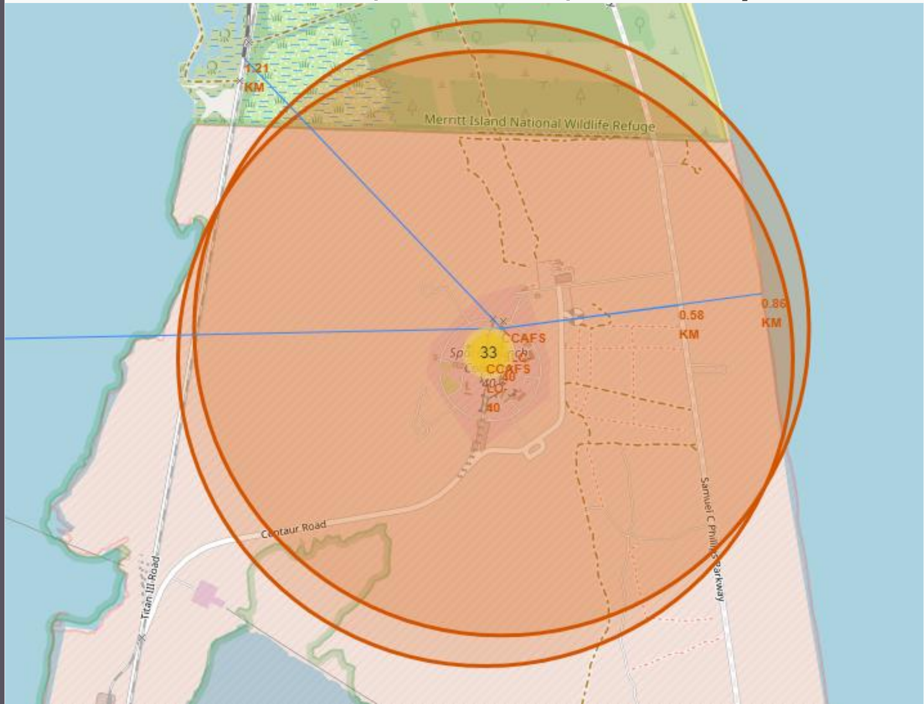
Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

Results

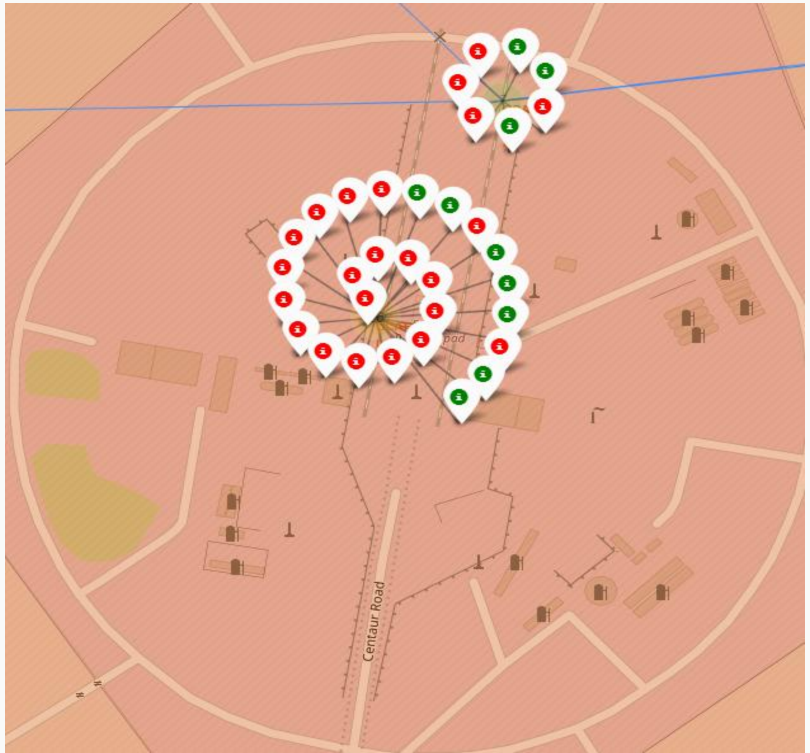
Exploratory Analysis: Interactive Map with Folium

Markers placed around Cape Canaveral Space Launch Complex in Florida.

Number of Launches shown in yellow circles. Blue lines showing distance to nearest coastline, railroad, and city.



Zooming in and clicking on yellow circles reveals pop-ups showing individual launches, colored for Landing Success or Failure.



Results

Predictive Analysis: Scikit-learn Machine Learning Models

Training Set: 72 records

Testing Set: 18 records

For each model, created a GridSearchCV object to test multiple hyper-parameters and return the best scoring version on the training data.

Then, checked how well the model predicted the testing data.

Each model was a classification model, attempting to predict Landing Success or Failure based on all of the selected features.

Results

Predictive Analysis: Logistic Regression Model

GridSearchCV

best_estimator_: LogisticRegression

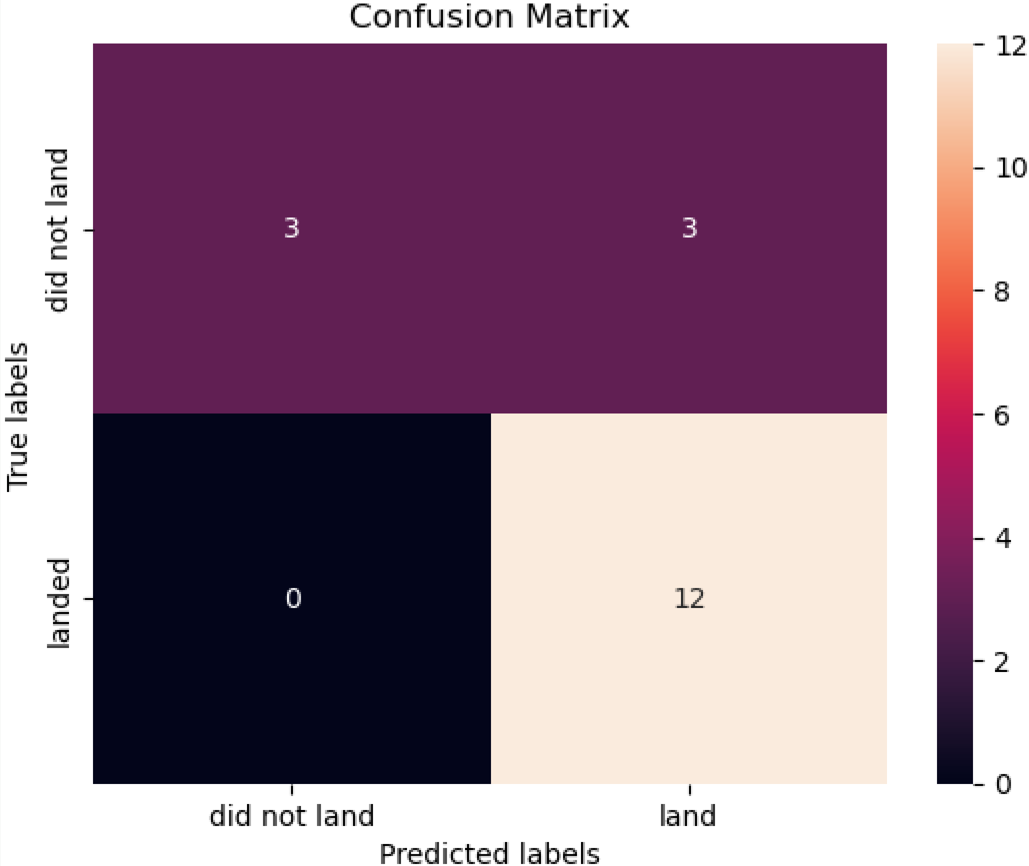
LogisticRegression

LogisticRegression(C=0.01)

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
training accuracy : 0.8464285714285713
test set accuracy : 0.8333333333333334
```

True Postive - 12 (True label is landed, Predicted label is also landed)

False Postive - 3 (True label is not landed, Predicted label is landed)



Results

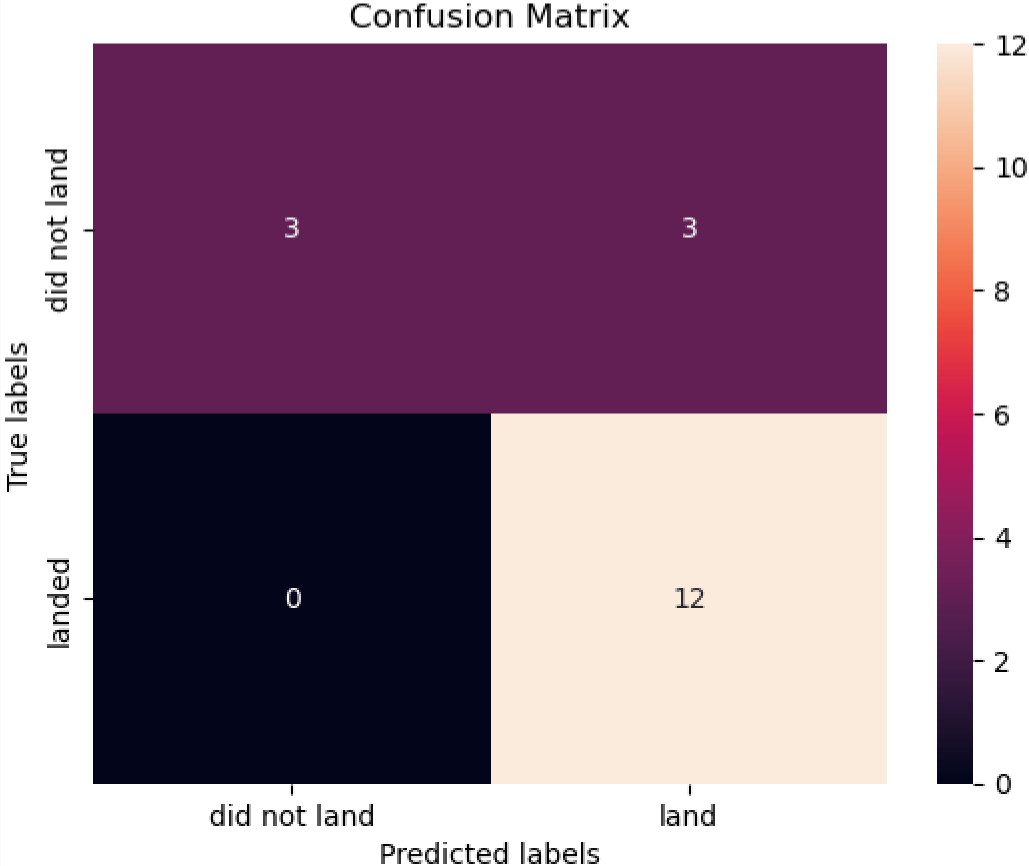
Predictive Analysis: Support Vector Machine Model

```
GridSearchCV
best_estimator_: SVC
SVC
SVC(gamma=0.03162277660168379, kernel='sigmoid')
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
training accuracy : 0.8482142857142856
test set accuracy : 0.8333333333333334
```

True Postive - 12 (True label is landed, Predicted label is also landed)

False Postive - 3 (True label is not landed, Predicted label is landed)



Results

Predictive Analysis: Decision Tree Classifier Model

GridSearchCV

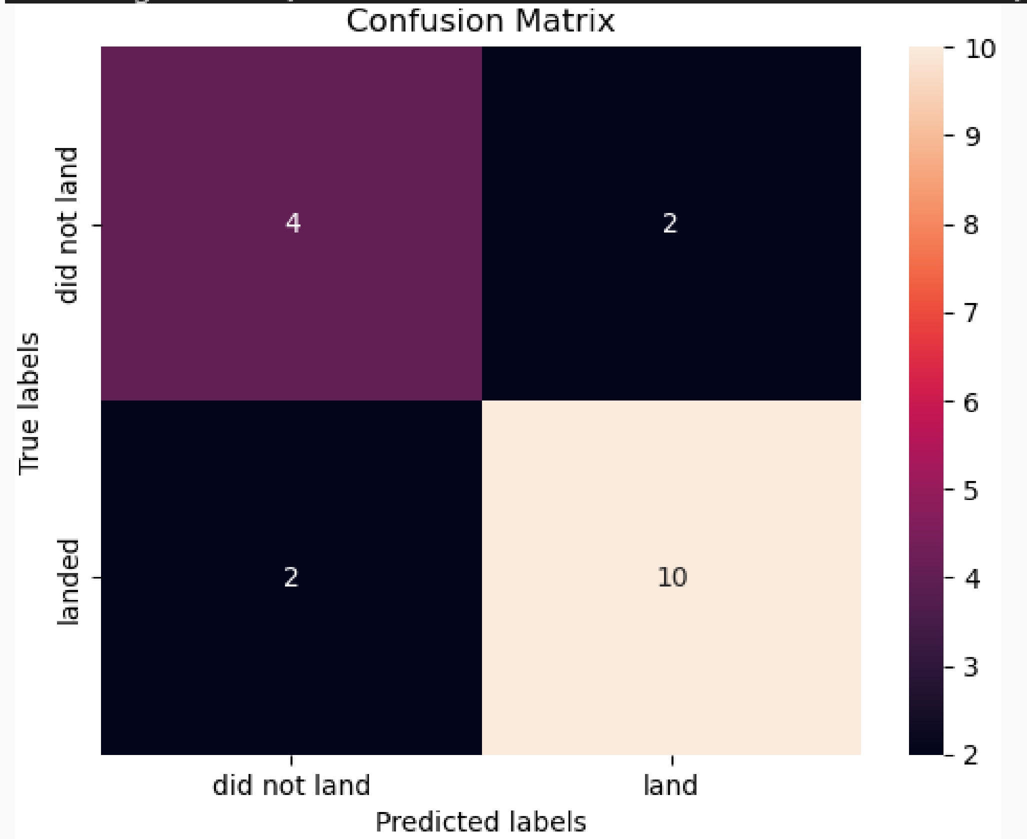
best_estimator_: DecisionTreeClassifier

DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=6, max_features='sqrt')
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt',
                                           'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
training accuracy : 0.875
test set accuracy : 0.7777777777777778
```

True Postive - 10 (True label is landed, Predicted label is also landed)
False Postive - 2 (True label is not landed, Predicted label is landed)
False Negative - 2 (True label is landed, Predicted label is not landed)



Results

Predictive Analysis: K Nearest Neighbors Model

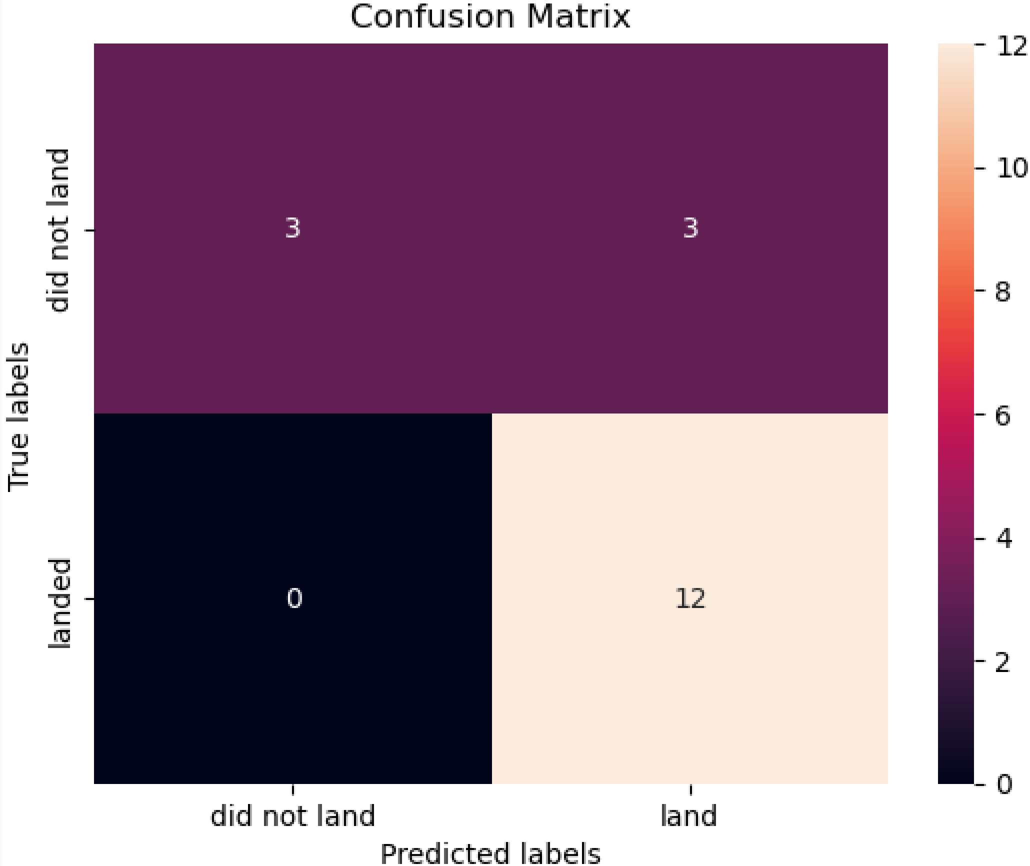
```
GridSearchCV
best_estimator_: KNeighborsClassifier
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10, p=1)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
training accuracy : 0.8482142857142858
```

```
test set accuracy : 0.8333333333333334
```

True Postive - 12 (True label is landed, Predicted label is also landed)

False Postive - 3 (True label is not landed, Predicted label is landed)



Results

Predictive Analysis: Scikit-learn Machine Learning Models

The models all performed somewhat similarly.

Their training accuracies varied, but on the training data, 3 of the 4 got the same results with 15 of 18 of the test records predicted correctly. The predictions that were incorrect came from predicting 3 launches that failed would've been successful.

The Decision Tree Classifier was inconsistent and predicted 13 of the 18 test records correctly, classifying 2 of the successful landings as failures.

Conclusions

- Landing success has increased over time and launch sites with more launch attempts have a higher landing success rate
- Vandenberg (CA) and Kennedy Space Launch (FL) Complexes successfully land more often than Cape Canaveral (FL) site
- ES-L1, GEO, HEO, SSO, VLEO Orbit Type launches successfully land more often than other orbits
- Higher Pay Load Mass doesn’t seem to negatively effect landing success

Conclusions

The predictive analysis using various machine learning models appears promising. Their accuracy is serviceable with a final data set of only 90 records and 18 of them used for testing.

Analysis of their performance as well as their reliability can be improved if we obtain more quality, relevant data to train and test on.

THANK YOU!

Cody Cline

cccline4@gmail.com