AI CAPSTONE PRESENTATION

SpaceX Data

Shayan Khan
2024-7-17

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

In this final capstone project of the IBM Data Science Course multiple machine learning classification algorithms were utilized to predict landing rates of the Falcon 9 first stage. The Falcon 9 was advertised on SpaceX's website. Multiple graphs as well as several classification algorithms are used to predict the landings of Spacex's Falcon 9

Multiple methodologies were used in this capstone project to analyze data, such as:

- Data collection & wrangling
- Interactive visual analytics
- Predictive analysis

The capstone consists of these main steps:

- Data collection & formating
- Exploratory data analysis
- Data visualization
- Machine learning prediction

Introduction

In this capstone project using the power of data science we will predict if the initial test launches of the Falcon 9 rocket by SpaceX will be successful or not. The objective of this capstone is to determine the viability of the rocket and it's chances of being successful for real launches.

Given a set of features about the rocket’s launches such as payload mass, orbit type, launch site, and booster version we can find out if the rocket is reliable or not.

Methodology

- Data collection, wrangling, and formatting
  - SpaceX API
  - Web Scraping
  - Beautiful Soup
- Exploratory data analysis
  - Pandas
  - Numpy
  - SQL
- Data visualization
  - Matplotlib
  - Seaborn
  - Folium
  - Dash
- Machine learning predictions
  - Logistic Regression
  - K-nearest Neighbors (KNN)
  - Support Vector Machines (SVM)
  - Decision Tree Classifier

Data Collection

- The data is retrieved from the SpaceX website, https://api.spacexdata.com/v4/launches/past

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

[7]: response = requests.get(spacex_url)
```

- After retrieving the data we turn the json file into a pandas dataframe
- The pandas dataframe has 94 rows and 17 columns
- Below is a picture of the first 5 columns of the dataframe

[38]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

- The below line of code is used to filter our the falcon 1 launches so that the data frame only shows falcon 9 launches

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

Data Collection

- To deal with the missing values or NaN's in the payload mass column the mean of the column is used to fill the NaN values.

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

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

- In the second data collection lab, data of the falcon 9 launches is web scraped from a wikipedia page via beautiful soup.

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
response = requests.get(static_url)
soup = BeautifulSoup(response.text, "html.parser")
```

- Using beautiful soup, column names, headers, and tables are extracted from the wikipedia page
- The extracted data is then turned into a pandas dataframe

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	F9 v1.0B0007.1	No attempt\n	1 March 2013	15:10
...	...	...	...	...	...	...	...	...	...	...	...
116	117	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1051.10	Success	9 May 2021	06:42
117	118	KSC	Starlink	~14,000 kg	LEO	SpaceX	Success\n	F9 B5B1058.8	Success	15 May 2021	22:56
118	119	CCSFS	Starlink	15,600 kg	LEO	SpaceX	Success\n	F9 B5B1063.2	Success	26 May 2021	18:59
119	120	KSC	SpaceX CRS-22	3,328 kg	LEO	NASA	Success\n	F9 B5B1067.1	Success	3 June 2021	17:29
120	121	CCSFS	SXM-8	7,000 kg	GTO	Sirius XM	Success\n	F9 B5	Success	6 June 2021	04:26

Data Wrangling

In this lab EDA was done to the SpaceX Falcon 9 dataframe to retrieve the following info

- Finding the number of launches on each site
- Calculating the number and occurrence of each orbit
- Calculating the number and occurence of mission outcome of the orbits
- Creating a landing outcome label from Outcome column

The main purpose of the lab was to find the outcome of each launch by separating the successful and unsuccessful launches.

At the end of the lab the below line of code was used to determine the success rate of the launches

We can use the following line of code to determine the success rate:

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

[26]: 0.6666666666666666
```

EDA With SQL

- The main purpose of this lab was to use SQL to perform EDA on the SpaceX dataset
- The following queries done to the Space X dataset include the following:
  - Displaying the names of the unique launch sites in the space mission
  - Displaying 5 records where launch sites begin with the string 'CCA'
  - Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - Displaying average payload mass carried by booster version F9 v1.1
  - Listing the date when the first successful landing outcome in ground pad was achieved.
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Listing the total number of successful and failure mission outcomes
  - Listing the names of the booster versions which have carried the maximum payload mass
  - Listing the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
  - Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

EDA With SQL

- The image below shows one of the tasks in this lab
- SQL is used to query the dataset

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

[54]:
```
%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) AS Landing_Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY  Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;
```

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

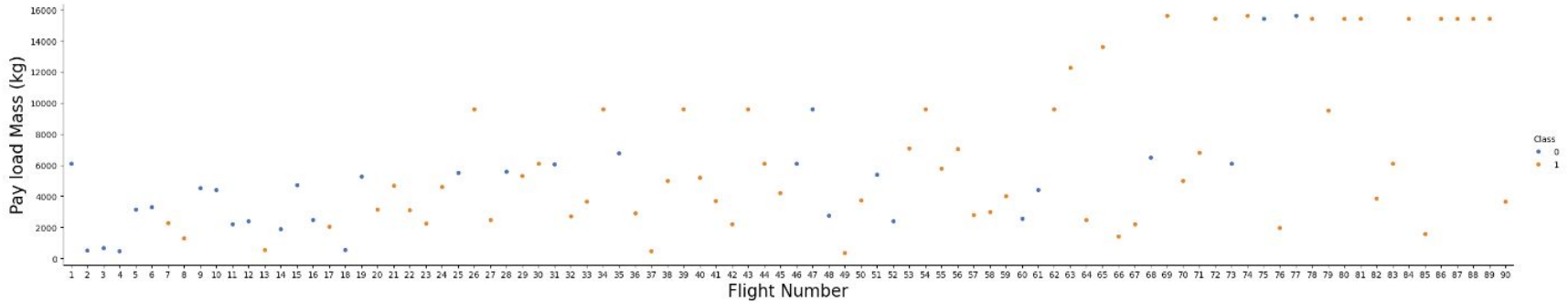
[54]:

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

EDA With Visualization (Matplotlib & Pandas)

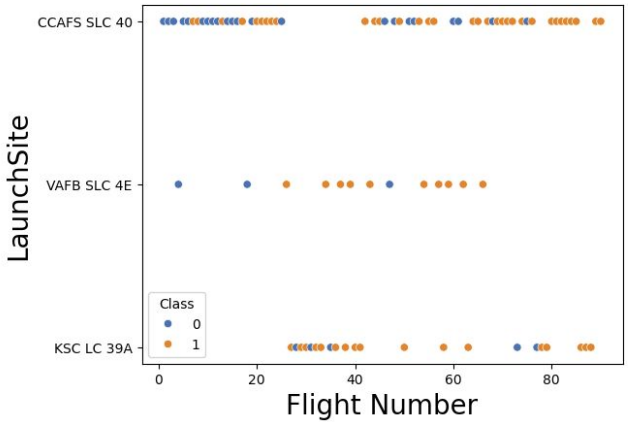
- In this lab exploratory data analysis and feature engineering is performed on the SpaceX dataset using Pandas and Matplotlib
- The following image shows the catplot of payload mass and flight number

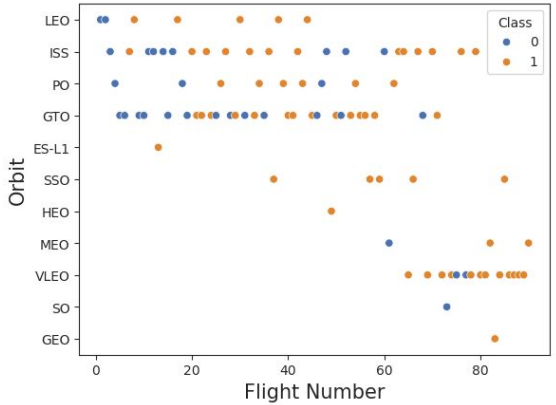
```
[5]: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
     plt.xlabel("Flight Number",fontsize=20)
     plt.ylabel("Pay load Mass (kg)",fontsize=20)
     plt.show()
```

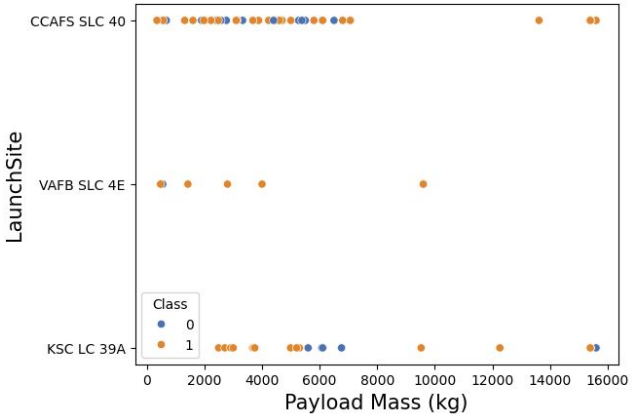


EDA With Visualization (Matplotlib & Pandas)

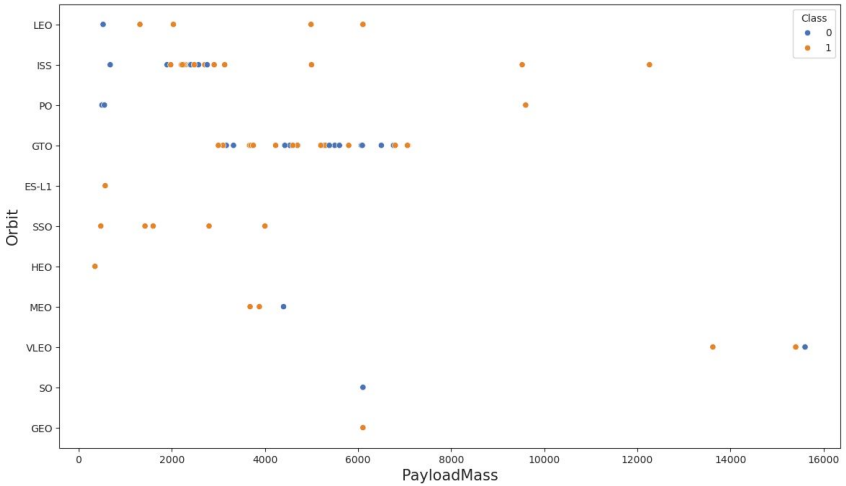
- Various graphs like the one on the previous slides are created to show the relationship between features in the dataset.
- The graphs are shown below

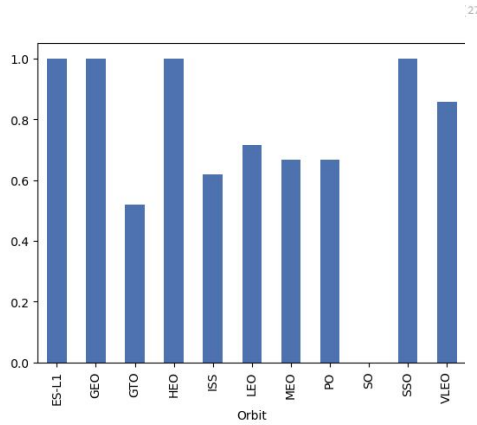


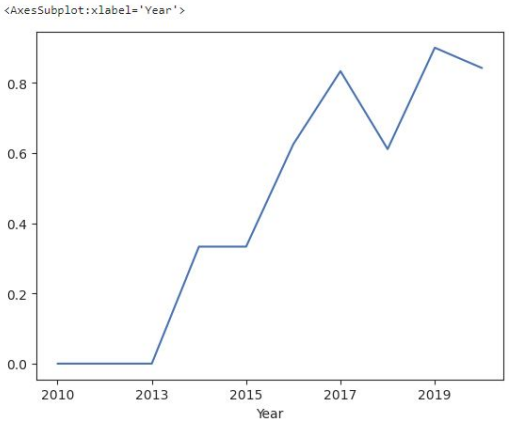




EDA With Visualization (Matplotlib & Pandas)







EDA With Visualization (Matplotlib & Pandas)

- At the end of the lab feature engineering is used to select the features that will be used in success prediction in the future module.

[28]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]
features.head()
```

[28]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

- One Hot Encoding is also used

[30]:
```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = features

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['Orbit'])], axis=1)
features_one_hot.drop(['Orbit'], axis = 1,inplace=True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['LaunchSite'])], axis=1)
features_one_hot.drop(['LaunchSite'], axis = 1,inplace=True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['LandingPad'])], axis=1)
features_one_hot.drop(['LandingPad'], axis = 1,inplace=True)

features_one_hot = pd.concat([features_one_hot,pd.get_dummies(df['Serial'])], axis=1)
features_one_hot.drop(['Serial'], axis = 1,inplace=True)

features_one_hot.head()
```

[30]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	ES-L1	GEO	...	B1048	B1049	B1050	B1051	B1054	B1056	B1058	B1059	B1060	B1062
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Visual Analytics with Folium

- In this lab more interactive and visual analytics are created with folium
- At the start of the lab, longitude and latitude coordinates are obtained of the launch sites

[6]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

[6]:

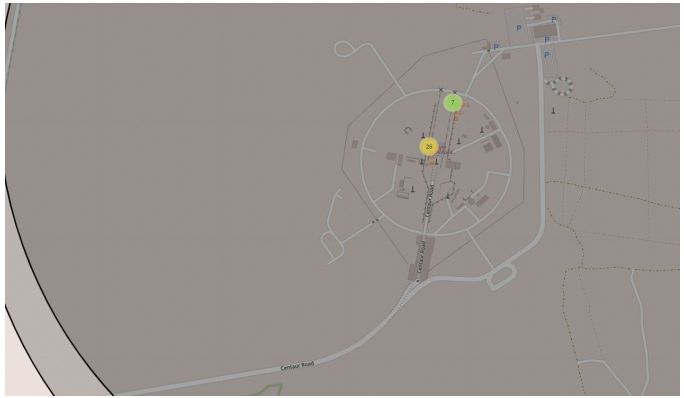
	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

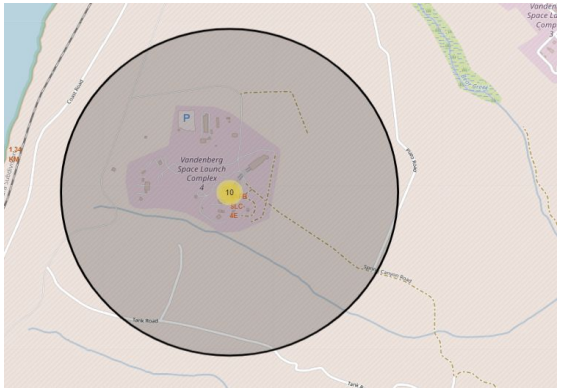
- Next a folium circle was used to highlight a specific area on the map

[8]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # 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>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Visual Analytics with Folium

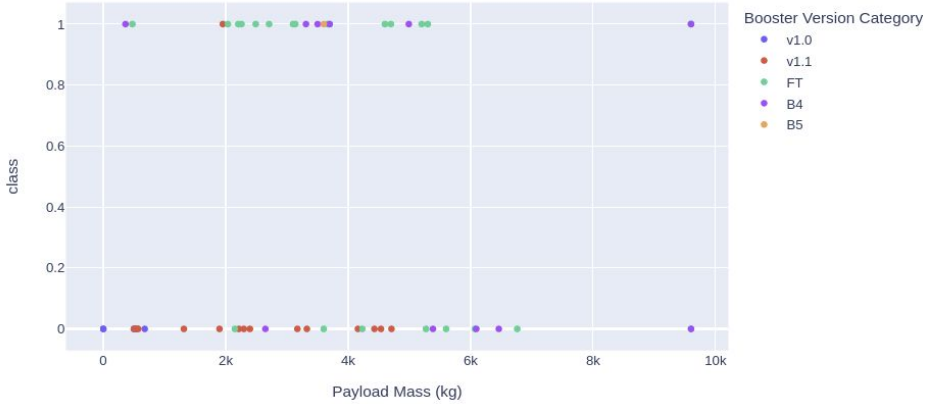
- The lab consisted of using the many features of folium to:
  - Add circles for each of the launch sites of falcon 9
  - Adding the launch outcomes for each site to see which sites have high success rates
  - Adding mark clusters for markers that are close to one another
  - Calculating the distances between a launch site to its proximities
  - Drawing a PolyLine between a launch site to the selected coastline point

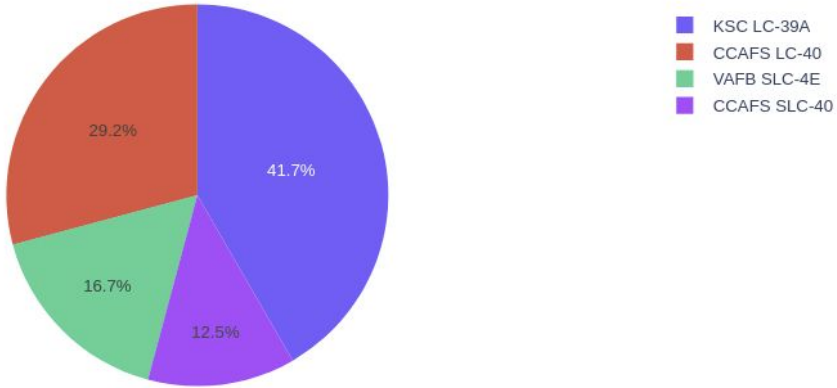




Interactive Dashboard with Plotly & Dash

- In this lab plotly and dash were used to create an interactive dashboard with a drop down menu and sliders, this allowed the user to toggle and change the input for the graph.
- 2 charts were used in this lab which include
  - Pie Chart
  - Scatter Plot
- The pie chart shows the total successful launches from each launch sites
- The scatter plot shows the correlation between payload mass and mission outcome for each launch site





Machine Learning Prediction

- In this lab several machine learning classification algorithms were used to find the success rate of the Falcon 9 Launches
- The algorithms used in this lab include:
  - K-nearest Neighbors
  - Decision Tree Classifier
  - Support Vector Machine (SVM)
  - Logistic Regression

- The lab started by creating the Y and X label and standardizing the X data
- Next the data was split into training (80%) and test (20%) data

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

Machine Learning Prediction

- Logistic Regression was then used

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

```
[27]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
      lr=LogisticRegression()

      logreg_cv = GridSearchCV(estimator=lr, cv=10, param_grid=parameters)
      logreg_cv.fit(X_train, Y_train)
```

```
[27]: ▸ GridSearchCV
      ▸ estimator: LogisticRegression
          ▸ LogisticRegression
```

- SVM was used next

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

```
33]: svm_cv = GridSearchCV(estimator=svm, cv=10, param_grid=parameters)
     svm_cv.fit(X_train, Y_train)
```

```
33]: ▸ GridSearchCV
     ▸ estimator: SVC
         ▸ SVC
```

```
34]: print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
     print("accuracy :",svm_cv.best_score_)
```

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

Machine Learning Prediction

- The Decision Tree Classifier was used next

```
[40]: 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()
```

```
[41]: tree_cv = GridSearchCV(estimator=tree, cv=10, param_grid=parameters)
      tree_cv.fit(X_train, Y_train)
```

- Finally KNN was used

```
[46]: 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()
```

```
[47]: knn_cv = GridSearchCV(estimator=KNN, cv=10, param_grid=parameters)
      knn_cv.fit(X_train, Y_train)
```

```
/lib/python3.11/site-packages/threadpoolctl.py:1019: RuntimeWarning: libc n
  warnings.warn(
```

```
[47]: GridSearchCV
      estimator: KNeighborsClassifier
          KNeighborsClassifier
```

Results

- The results of each of the algorithms are shown below

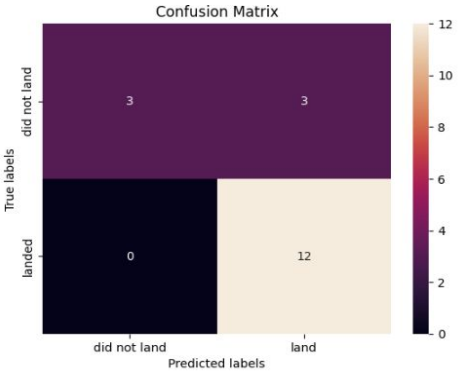
```
print("accuracy :",logreg_cv.score(X_test, Y_test)
accuracy : 0.8333333333333334
```

```
print("accuracy :", svm_cv.score(X_test, Y_test))
accuracy : 0.8333333333333334
```

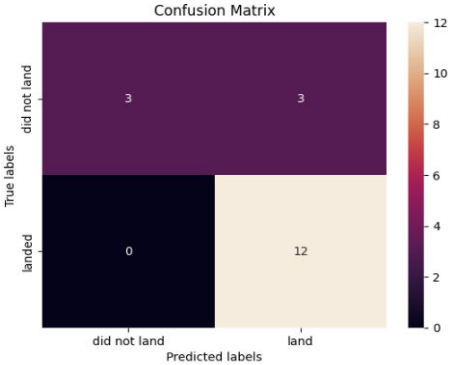
```
print("accuracy :", tree_cv.score(X_test, Y_test))
accuracy : 0.8888888888888888
```

```
print("accuracy :", knn_cv.score(X_test, Y_test))
accuracy : 0.8333333333333334
```

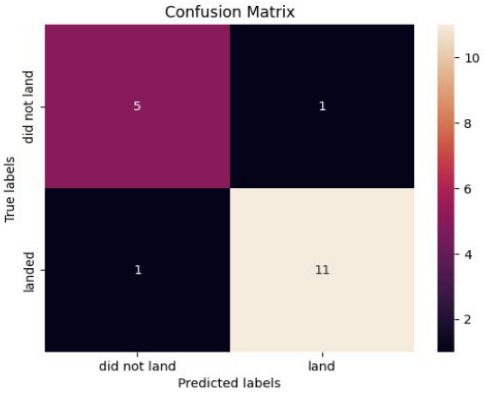
- Confusion Matrix for each of the models are shown below



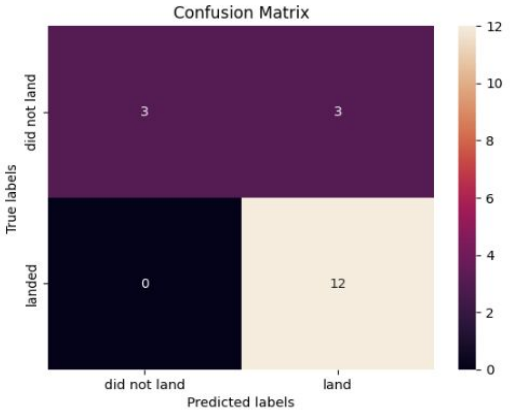
Logistic Regression



SVM



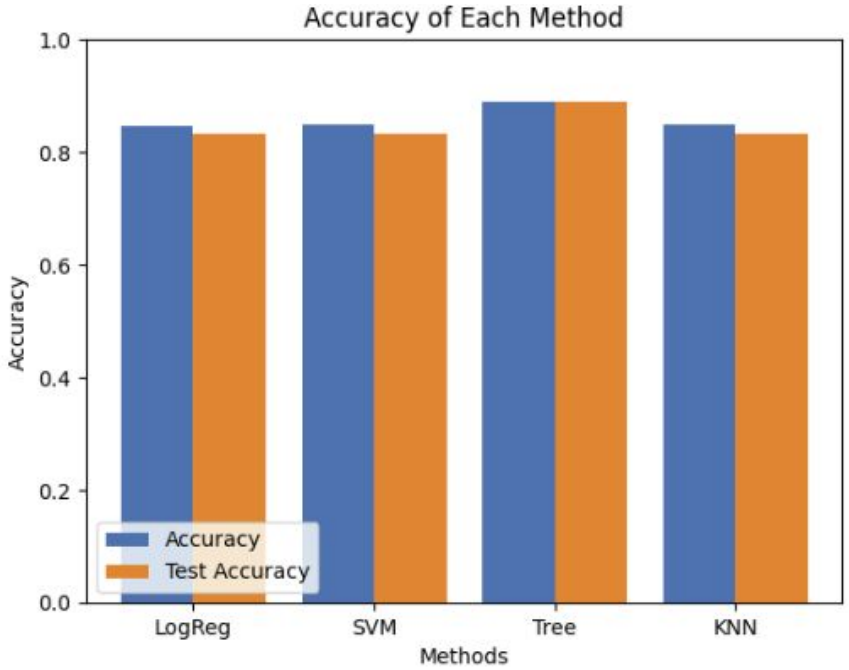
Decision Tree



KNN

Results

- From the chart below we can see that the decision tree classifier is the most accurate model for predicting successful landings, with training and test result of 89 and 90 respectfully



Conclusion

This AI Capstone project has truly been a wonderful and knowledgeable journey, by helping me refresh my memory on the many lessons, videos, and labs done in this course. By combining everything I learned in this compact data science course I finished the capstone project. From using machine learning models, building graphs, creating dashboards, and handling data with numpy and & pandas this course has taught me a lot. By using various different python modules a lot of testing and exploring was done on the SpaceX Falcon 9 data allowing us to get more information on it's launches, launch outcomes, landing area, boosters, orbits, etc.