IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

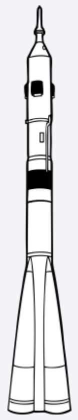
Ellei Shamaev
1 Oct, 2023

Outline

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

Executive Summary

- Features analysis and Interactive analytics
  - included for better understanding
- Predictive analysis results
  - Discovered the model exhibited the highest accuracy.
  - Accuracy is dependent on the training seed, suggesting a small dataset size.
  - Notably, some important data (e.g., wind, clouds, SpaceX AI versioning) was excluded from the analysis.

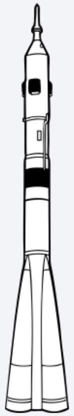


Section 1

Methodology

Introduction

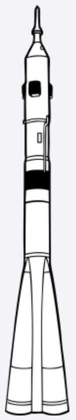
- SpaceX, a spacecraft manufacturer and launcher, is changing the world by providing access to the Internet around the world. The key to its success is a cost-effective method for placing satellites into space orbits.
- We conduct exploratory data analysis **to gain insights** from SpaceX launch data.



Methodology

Executive Summary

- **Data collection** — SpaceX API, web scraping
- **Data wrangling** — Discovery, transformation, validation
- **Exploratory data analysis** — Visualization
    - **Interactive visual analytics** — Folium and Plotly Dash
- **Predictive analysis results** — Achieved > 84% accuracy

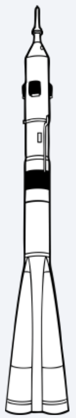


Data Collection – SpaceX API

- API SpaceX
- Available Features
  - Date, Time, Version, Launch Site,
  - Payload info, Customer,
  - Orbit, Launch outcome, Landing
- Missing values
  - Replace with the average value or ignore it?

```
df.head()
```

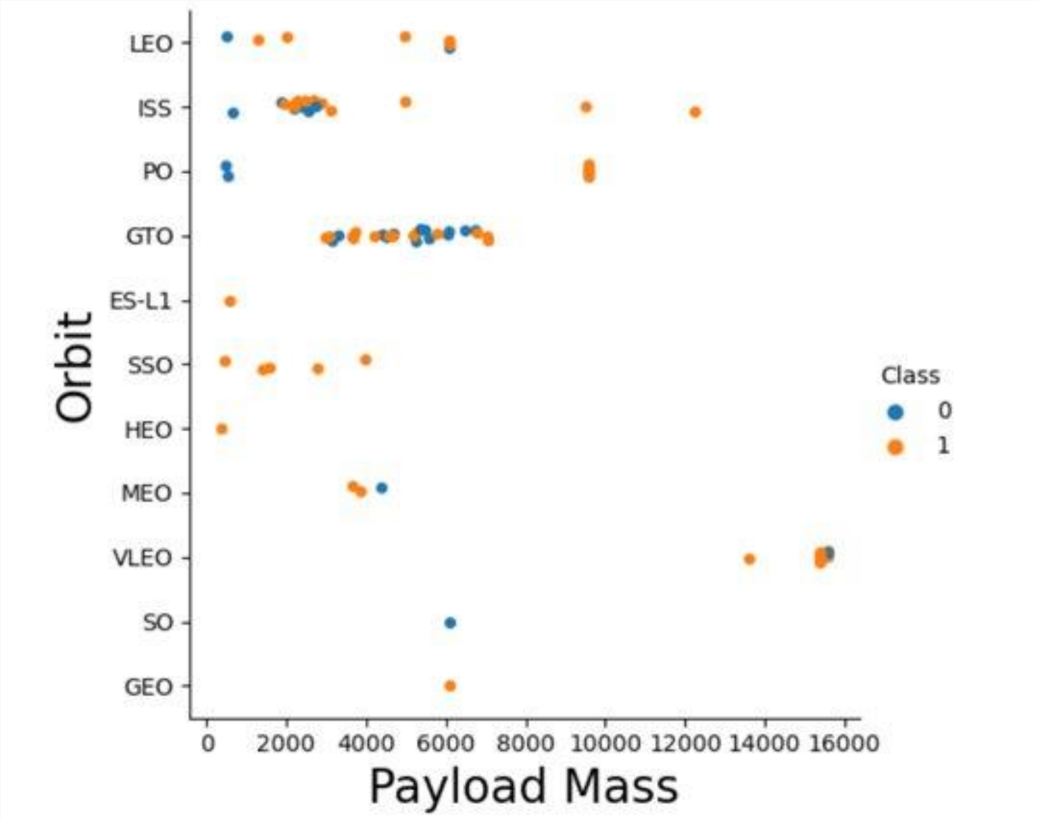
	Flight No.	Date	Time	Version Booster	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Booster landing
0	[4 June 2010,, 18:45]	4 June 2010	18:45	F9 v1.0B0003.1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	Failure
1	[8 December 2010,, 15:43]	8 December 2010	15:43	F9 v1.0B0004.1	CCAFS	Dragon	0	LEO	NASA	Success	Failure
2	[22 May 2012,, 07:44]	22 May 2012	07:44	F9 v1.0B0005.1	CCAFS	Dragon	525 kg	LEO	NASA	Success	No attempt\n
3	[8 October 2012,, 00:35]	8 October 2012	00:35	F9 v1.0B0006.1	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	No attempt
4	[1 March 2013,, 15:10]	1 March 2013	15:10	F9 v1.0B0007.1	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success\n	No attempt\n



The related code is published at github.com/elleish/IBMDS-Final-Project/

Data Collection – Scraping

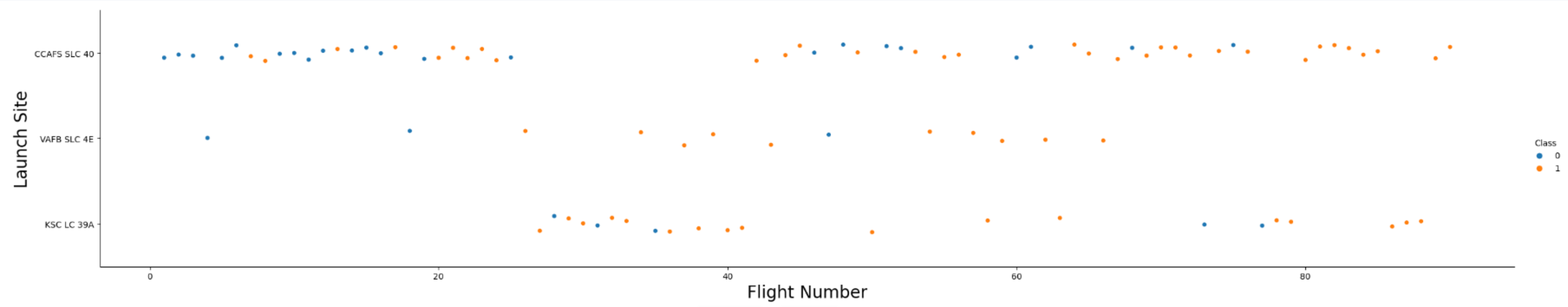
- **Web scraping**
  - HTML → *BeautifulSoup* → JSON
  - JSON → *the algorithm* → Table
- Alternative web scraping
  - HTML → *pandas.read_html* → Table
- Missing values
  - Replace with the average value or ignore it?

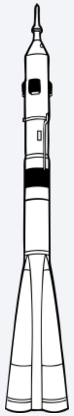


The related code is published at github.com/elleish/IBMDS-Final-Project/

Data Wrangling

- Discovery
  - Context
  - Planning
- Transformation
  - Structuring Data
  - Normalizing Data
  - Cleaning Data
  - Enriching Data
- Validation
  - Consistency
  - Quality

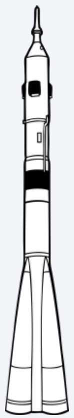




The related code is published at github.com/elleish/IBMDS-Final-Project/

EDA with Data Visualization

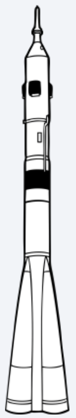
- Exploration
  - Different launch sites, orbits, and payload masses have different success rates.
  - The success rate is increasing.
  - With heavy payloads, the rate of successful or positive landings is higher for Polar, LEO, and ISS orbits.
  - Additional insights are obtained by creating scatter plots that depict the relationship between two variables, with successful outcomes shown in orange and unsuccessful outcomes in blue.



The related code is published at github.com/elleish/IBMDS-Final-Project/

EDA with SQL

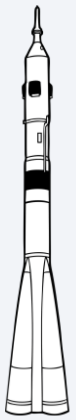
- Basic Queries of SQL
  - select distinct values from the table – select distinct *Value* from *Table*
  - select the sub-table from the table with the condition – select * from *Table* where *Condition*
  - sum values from the sub-table queried by the condition – select *sum*(*Value*) as *SumOfValue* from *Table* where *Condition*
  - Instead of the function *sum* other aggregating functions *avg (average), min*, *max*, *count*, *count distinct* can be used
  - group the table by the value 1 – select *Value1, sum(Value2)* from *Table* group by *Value1*



The related code is published at github.com/elleish/IBMDS-Final-Project/

EDA with SQL

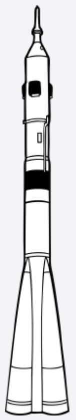
- Examples of Logical Conditions in SQL
  - *Value1* = 1
  - (*Value1* > 1 and *Value2* < 1) or *Value3* != 3
  - *Value1* between 1 and 3
  - *Value1* = (select *Value2* from *Table2*)
  - *Value2* = 'SpaceX 9'
  - *Value2* => 'a' and *Value2* <='z'
  - lower(*Value2*) in ("Falcon9 v1.0", "Falcon9 v2.0")



The related code is published at github.com/elleish/IBMDS-Final-Project/

Build a Dashboard with Plotly Dash

- Explain why you added those plots and interactions
- Interactive visualizations change the graph in response to certain events (e.g., mouse over, mouse click).
- Interactive visualization is implemented using a Python feature called callbacks.
- Callbacks allow functions to be called in response to system events.
- Interactive visualization is not possible in PowerPoint.



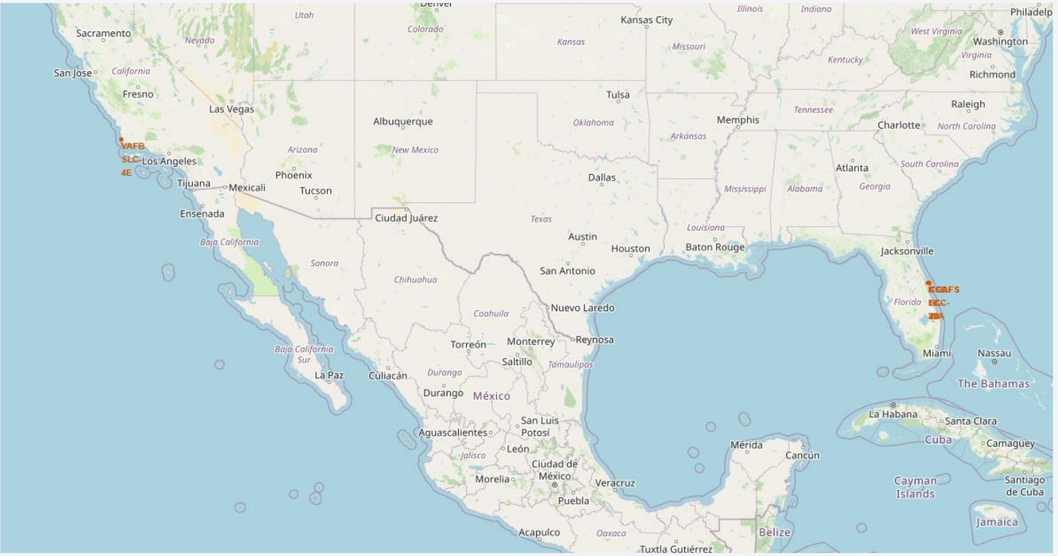
The related code is published at github.com/elleish/IBMDS-Final-Project/

Build an Interactive Map with Folium

- Each Launch site marked using folium.Circle folium.Marker
- Launch sites with the same Lat and Long are visualized using MarkerCluster()
- Path plotted using folium.PolyLine

	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

Folium

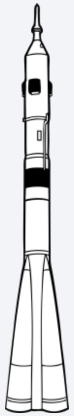


The related code is published at github.com/elleish/IBMDS-Final-Project/

Predictive Analysis (Classification)

- Data transformed using the standard scaler.
- The data is split into training and testing sets in an 80% to 20% proportion.
- Models including LogisticRegression, SVC, DecisionTreeClassifier, and KNeighborsClassifier from sklearn are trained on both the training and testing data.
- Optimal hyperparameters are determined using grid search (brute force).
- The DecisionTreeClassifier model has shown the best results.

Model	Accuracy
LogisticRegression	83.4%
SVC with sigmoid kernel	83.3%
DecisionTreeClassifier	87.7%
KNeighboursClassifier	84.8%

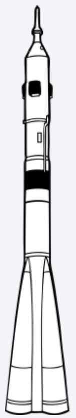


The related code is published at github.com/elleish/IBMDS-Final-Project/

Predictive Analysis (Classificator example)

- The model checks for 'Legs' first, and then 'ReusedCount.'
- If a rocket has not been reused yet, the most important features for predictive analysis are 'PayloadMass,' 'Orbit,' and 'Serial.'
- If a rocket has been reused, the most important features for predictive analysis are 'LaunchSite,' 'Orbit,' 'ReusedCount,' and 'Serial.'
- This model achieves an accuracy of 94%.

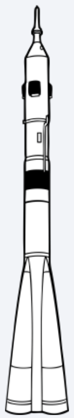
```
|--- Legs = True
|   |--- ReusedCount = 0
|   |   |--- PayloadMass <= 4983.50
|   |   |   |--- Orbit = ISS
|   |   |   |   |--- Serial != B1017
|   |   |   |   |   |--- class: Success
|   |   |   |   |--- Serial = B1017
|   |   |   |   |   |--- class: Crush
|   |   |   |--- Orbit = ISS
|   |   |   |   |--- FlightNumber <= 11
|   |   |   |   |   |--- class: Success
|   |   |   |   |--- FlightNumber >  11
|   |   |   |   |   |--- class: Crush
|   |   |--- PayloadMass >  4983.50
|   |   |   |--- class: Crush
|   |--- ReusedCount >=  1
|   |   |--- Serial != B1041
|   |   |   |--- Flights <= 4
|   |   |   |   |--- Serial != B1039
|   |   |   |   |   |--- class: Success
|   |   |   |   |--- Serial = B1039
|   |   |   |   |   |--- LaunchSite != KSC LC 39A
|   |   |   |   |   |   |--- class: Crush
|   |   |   |   |   |--- LaunchSite_KSC = LC 39A
|   |   |   |   |   |   |--- class: Success
|   |   |   |--- Flights >=  5
|   |   |   |   |--- ReusedCount <= 4
|   |   |   |   |   |--- class: Crush
|   |   |   |   |--- ReusedCount >= 5
|   |   |   |   |   |--- class: Success
|   |   |--- Serial = B1041
|   |   |   |--- Flights <= 1
|   |   |   |   |--- class: Success
|   |   |   |--- Flights >= 2
|   |   |   |   |--- class: Crash
|--- Legs = False
```



The related code is published at github.com/elleish/IBMDS-Final-Project/

Results

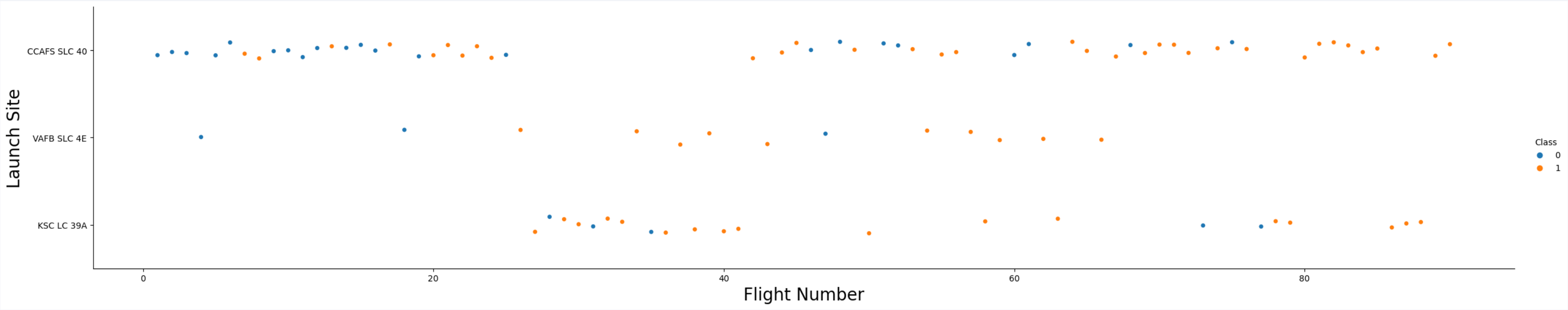
- Analysis of feature pairs:
  - checked the data for missing values and obvious errors
  - discovered some correlations
- Interactive analytics demo
  - included for better understanding
- Predictive analysis results
  - The decision tree model exhibited the highest accuracy.
  - Accuracy is dependent on the training seed, suggesting a small dataset size.
  - Notably, some important data (e.g., wind, clouds, SpaceX AI versioning) was excluded from the analysis.



Section 2

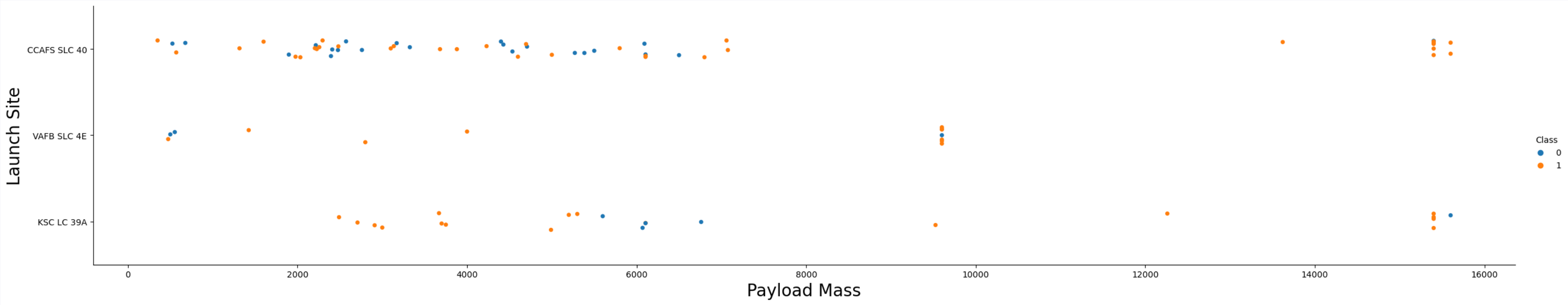
Insights drawn from EDA

Flight Number vs. Launch Site



- Different launch sites have different success rates.
- Launch site CCAFS SLC 40 is the most popular, with 9 successful launches.
- Launch site VAFB SLC 4E is the least popular, with 5 successful launches.
- The last 14 launches were all successful.

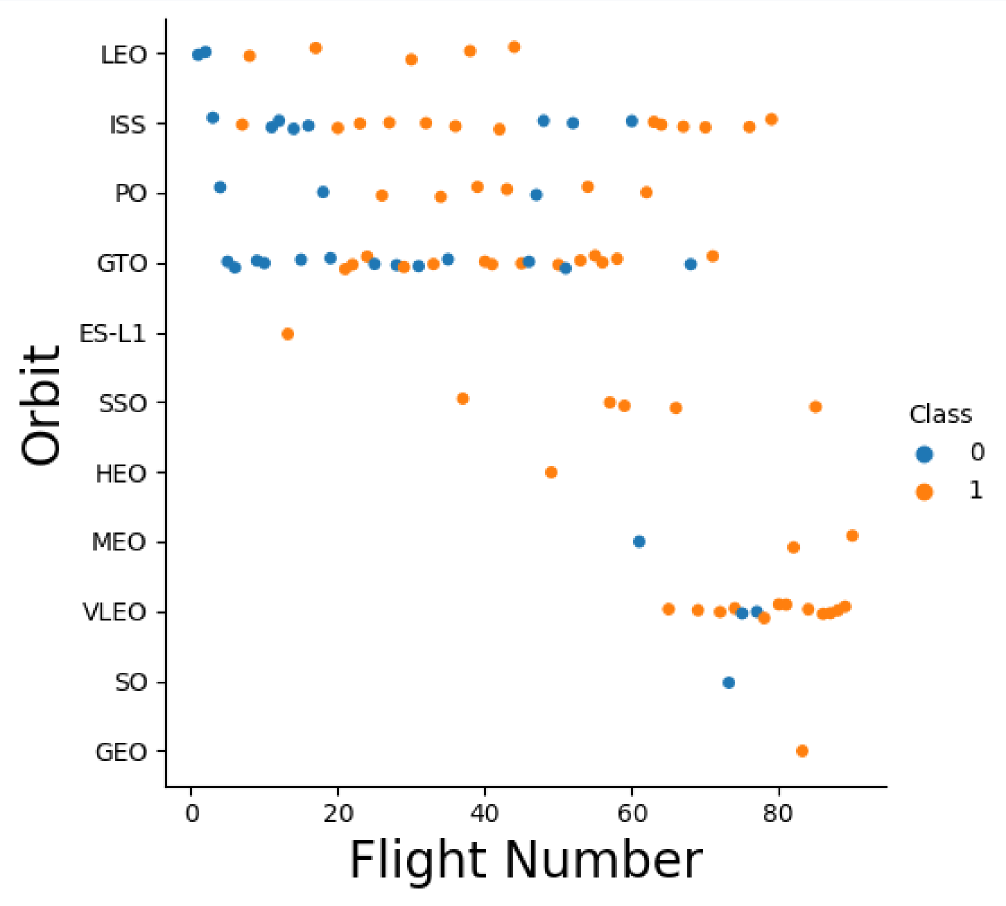
Payload vs. Launch Site



- The success rate at CCAFS SLC 40 Launch Site is high for launches with a high payload mass.
- VAFB SLC 4E and KSC LC 39A have varying success rates for different payload mass ranges.
- It may depend on latent factors that we do not know on this stage of research.

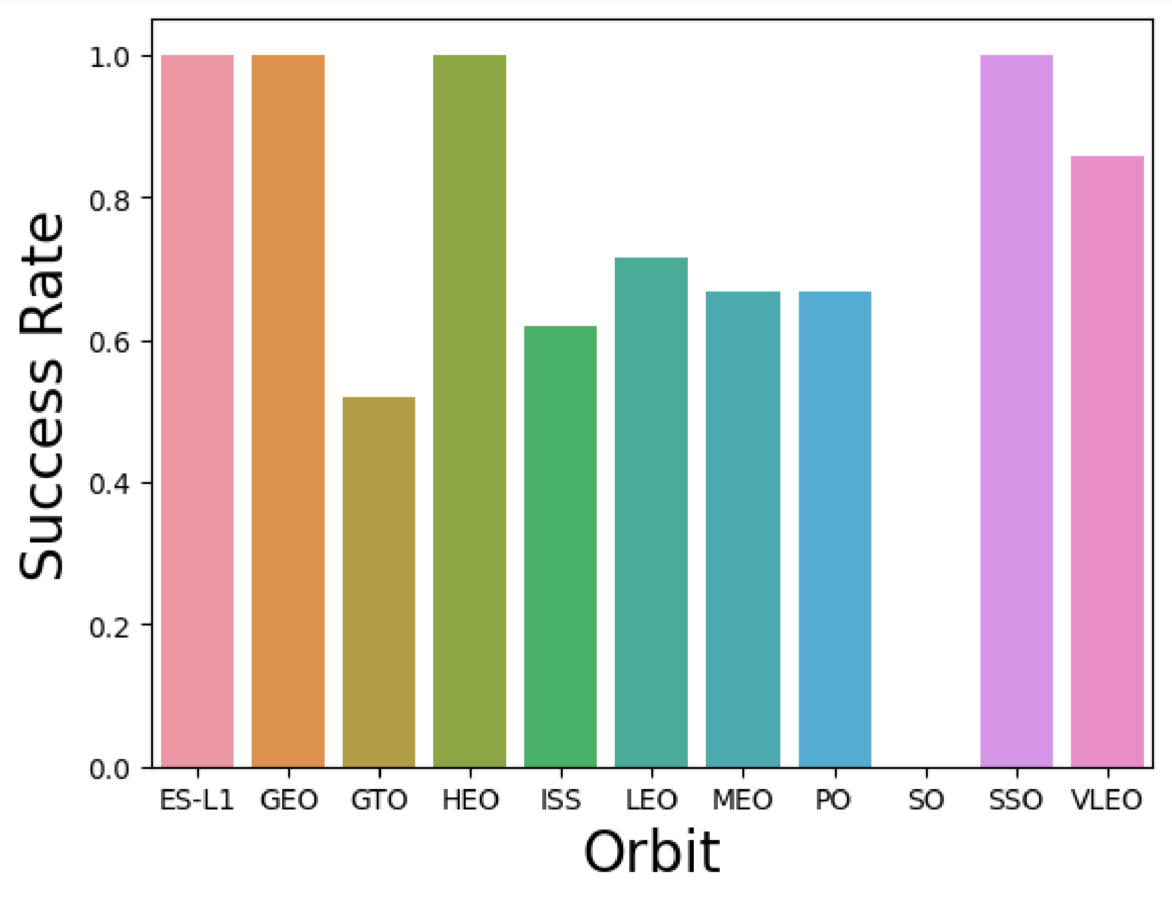
Flight Number vs. Orbit Type

- Launches to the GTO, SO orbits have the lowest success rate.
- Launches to the SSO, GEO, ES-L1, HEO orbit have a 100% success rate.
    - There was only one launch for each GEO, ES-L1, and HEO orbit



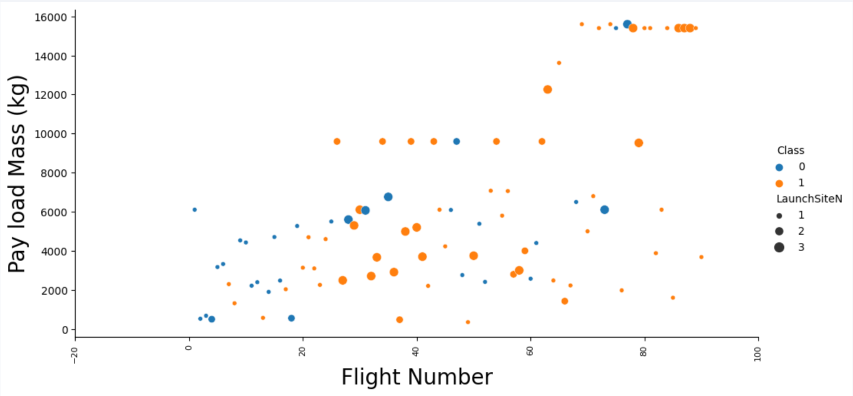
Success Rate vs. Orbit Type

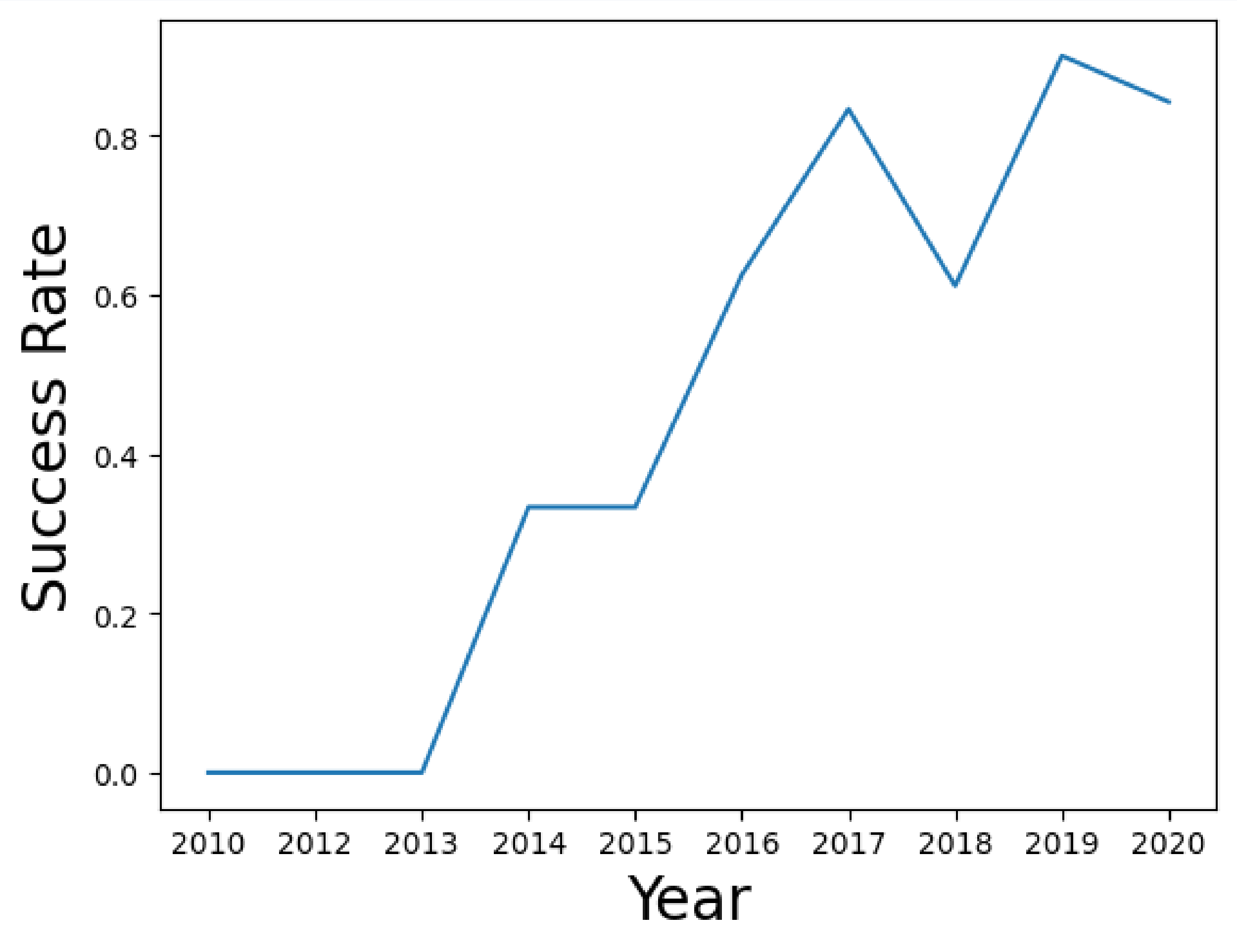
- Launches to the GTO, SO orbits have the lowest success rate.
- Launches to the SSO, GEO, ES-L1, HEO orbits have a 100% success rate



Launch Success Yearly Trend

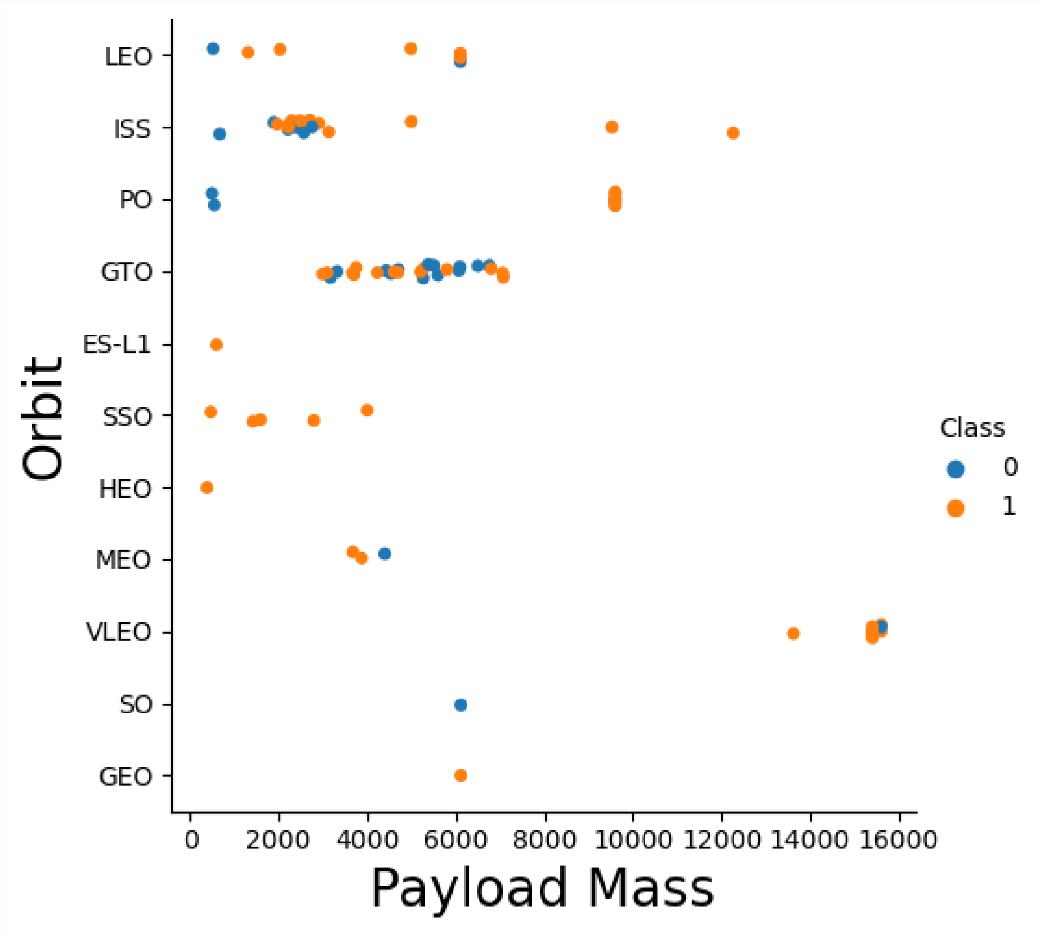
- The success rate is increasing.
- The reasons for the success rate drop in 2018 can be interesting.





Payload vs. Orbit Type

- With heavy payloads, the success rate for positive landings is higher for Polar, LEO, and ISS orbits.
- Payload mass was correlated with success rate before 2019.
- Payload mass did not correlate with the success rate after 2019.



All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

```
%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

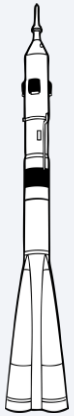
Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here

```
%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	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt



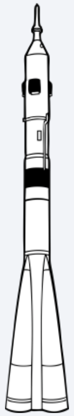
Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

```
%sql select sum(PAYLOAD_MASS__KG_) as Total_payload_mass from SPACEXTBL where lower(Customer) in ("nasa (crs)","nasa(crs)")
```

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

Total_payload_mass
45596



Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

```
%sql select avg(PAYLOAD_MASS__KG_) as avg_payload_mass_f9v1_1 from SPACEXTBL where lower(Landing_Outcome) in ("f9 v1.1", "f
```

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

avg_payload_mass_f9v1_1
2928.4

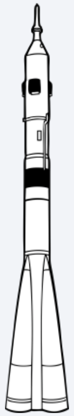
First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here

```
%%sql
select min(Date) as first_landing_ground_pad from SPACEXTBL
where Landing_Outcome = "Success (ground pad)"
```

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

first_landing_ground_pad
2015-12-22



Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Present your query result with a short explanation here

```
%%sql
select min(Date) as first_landing_ground_pad from SPACEXTBL
where Landing_Outcome = "Success (drone ship)"
  and PAYLOAD_MASS__KG_ > 4000
  and PAYLOAD_MASS__KG_ < 6000
```

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

first_landing_ground_pad
2016-06-05

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

```
%%sql
select Booster_Version from SPACEXTBL
where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
order by Booster_Version
```

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

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

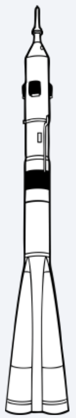
Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

```
%%sql
select count(*) as  total_success_fail_outcomes from SPACEXTBL
where Landing_Outcome like "Success%"
   or Landing_Outcome like "Failure%"
```

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

total_success_fail_outcomes
71



2015 Launch Records

- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Present your query result with a short explanation here

```
%%sql
select Date, Landing_Outcome, Booster_Version, Launch_Site
from SPACEXTBL
where Landing_Outcome = "Failure (drone ship)"
  and Date>='2015-01-01'
  and Date<='2015-12-31'
```

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

Date	Landing_Outcome	Booster_Version	Launch_Site
2015-10-01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

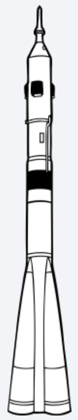
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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
- Present your query result with a short explanation here

```
%%sql
select  Landing_Outcome, count(*) as count_
from SPACEXTBL
where Landing_Outcome in ("Failure (drone ship)", "Success (ground pad)")
  and Date>='2010-06-04'
  and Date<='2017-03-20'
group by Landing_Outcome
```

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

Landing_Outcome	count_
Failure (drone ship)	5
Success (ground pad)	5



Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

- Intentionally blank page

Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Intentionally blank page

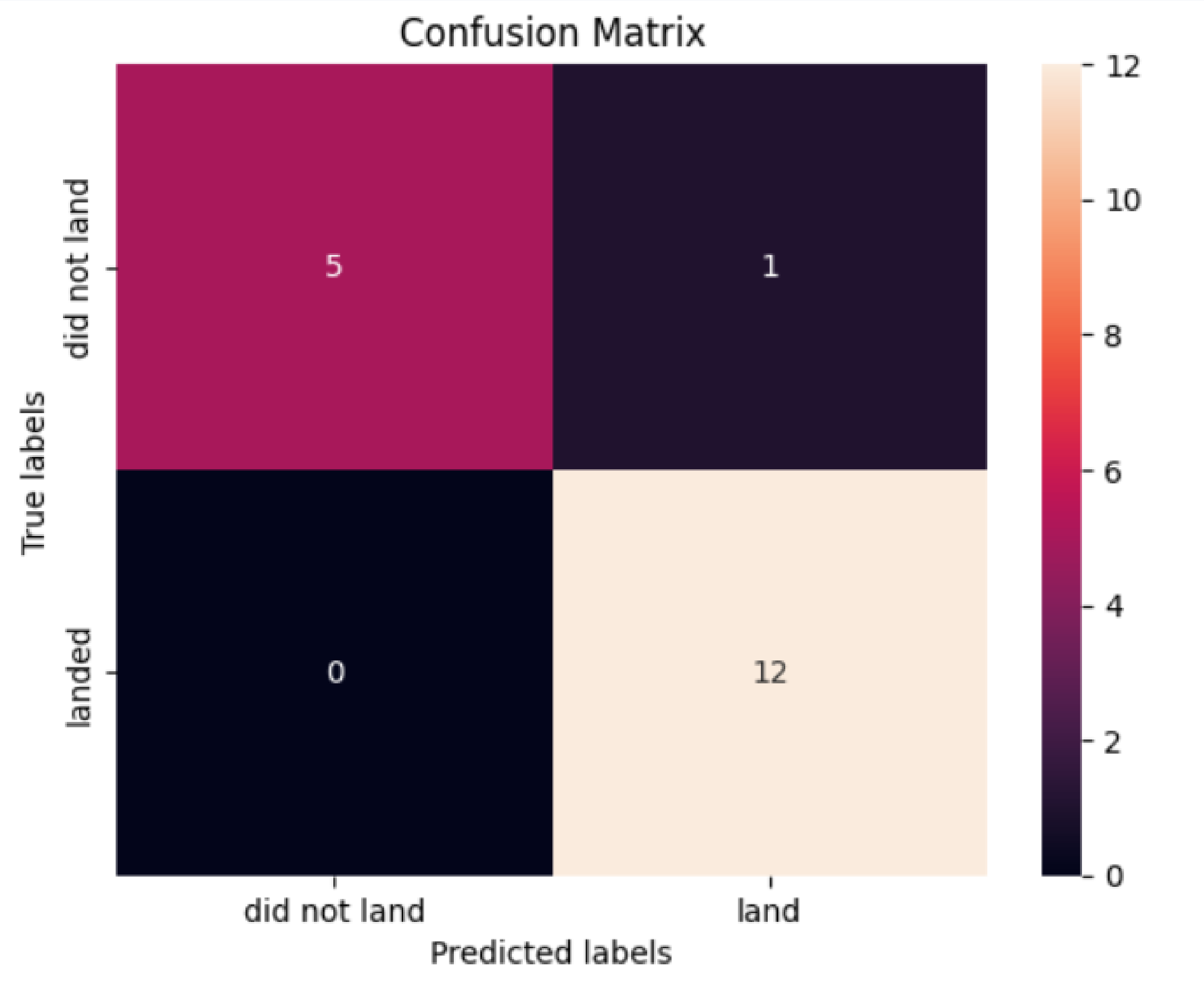
Section 5

Predictive Analysis (Classification)

Classification Accuracy

Model	Accuracy
LogisticRegression	83.4%
SVC with sigmoid kernel	83.3%
DecisionTreeClassifier	87.7%
KNeighboursClassifier	84.8%

Confusion Matrix



Appendix

- Intentionally blank page

Thank you!