Data-driven insights & predictive analysis

IBM Data Science Professional Certificate
Applied Data Science Capstone

Prepared by: Pawel Banasiewicz | Date: 2026-01-12

Executive Summary

What we built and what we found

- Goal: predict whether Falcon 9 first stage landing will be successful (Class = 1).
- Data: SpaceX API + Wikipedia launch tables + IBM-provided datasets for consistency.
- Approach: cleaning & feature engineering, EDA (visual + SQL), interactive analytics (Folium + Dash), modeling.
- Key pattern: success rate increases over time (FlightNumber / Year).
- Best model: SVM selected (tie on test accuracy resolved by higher CV score).

Introduction

Problem statement and target definition

- Business context: reusable first-stage landings lower launch costs and increase cadence.
- Task: binary classification - will the first stage land successfully?
- Label (Class): 1 = successful landing, 0 = failed / no successful landing (per capstone definition).
- Features include mission profile, payload mass, orbit, launch site, booster & re-use indicators.

```
[15]: data[['FlightNumber','PayloadMass','Orbit','LaunchSite','Outcome','Class']].head(10)
```

[15]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Outcome	Class
0	1	6123.547647	LEO	CCSFS SLC 40	None None	0
1	2	525.000000	LEO	CCSFS SLC 40	None None	0
2	3	677.000000	ISS	CCSFS SLC 40	None None	0
3	4	500.000000	PO	VAFB SLC 4E	False Ocean	0
4	5	3170.000000	GTO	CCSFS SLC 40	None None	0
5	6	3325.000000	GTO	CCSFS SLC 40	None None	0
6	7	2296.000000	ISS	CCSFS SLC 40	True Ocean	1
7	8	1316.000000	LEO	CCSFS SLC 40	True Ocean	1
8	9	4535.000000	GTO	CCSFS SLC 40	None None	0
9	10	4428.000000	GTO	CCSFS SLC 40	None None	0

```
[8]: landing_outcomes=data['Outcome'].value_counts()
     landing_outcomes
```

```
[8]: Outcome
     True ASDS      41
     None None      19
     True RTLS      14
     False ASDS      6
     True Ocean      5
     False Ocean     2
     None ASDS       2
     False RTLS      1
     Name: count, dtype: int64
```

```
[11]: landing_class = [0 if outcome in bad_outcomes else 1 for outcome in data['Outcome']]
      landing_class[:10], len(landing_class)
```

```
[11]: ([0, 0, 0, 0, 0, 0, 1, 1, 0, 0], 90)
```

This variable will represent the classification variable that represents the outcome of each launch. I
means the first stage landed Successfully

```
[12]: data['Class'] = landing_class
      data['Class'].value_counts()
```

```
[12]: Class
      1    60
      0    30
      Name: count, dtype: int64
```

Data Collection

SpaceX API + enrichment

- Pulled launch records from SpaceX API (launches/past endpoint).
- Enriched records via secondary endpoints: rockets, launchpads, payloads, cores.
- Filtered to Falcon 9 and a consistent date range used by the course labs.
- Exported intermediate dataset to CSV for downstream steps.

[38]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

[38]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
0	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
1	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
2	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
3	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
4	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
85	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True
86	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True
87	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True
88	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True
89	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True

Data Collection

Web scraping Wikipedia launch tables

- Used Wikipedia page (specific revision) to extract launch records from HTML tables.
- Handled noisy formatting: references, missing values, mixed text/link cells.
- Standardized fields (date/time, booster version, payload mass, orbit, landing outcome).
- Saved scraped dataset for comparison/EDA.

```
[33]: df.head()
```

[33]:

	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	F9 v1.07B0003.18	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.07B0005.18	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.07B0006.18	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.07B0007.18	No attempt\n	1 March 2013	15:10

Data Wrangling & Feature Engineering

Cleaning, labels, and one-hot encoding

- Created label Class from Outcome (success vs bad outcomes).
- Handled missing values (e.g., payload mass), standardized numeric types.
- Selected modeling features (numeric + categorical).
- Applied one-hot encoding to: Orbit, LaunchSite, LandingPad, Serial.
- Standardized features using StandardScaler before modeling.

```
[6]: data.isnull().sum()/len(data)*100

[6]: FlightNumber      0.000000
     Date              0.000000
     BoosterVersion    0.000000
     PayloadMass       0.000000
     Orbit             0.000000
     LaunchSite        0.000000
     Outcome           0.000000
     Flights           0.000000
     GridFins          0.000000
     Reused            0.000000
     Legs              0.000000
     LandingPad       28.888889
     Block             0.000000
     ReusedCount       0.000000
     Serial            0.000000
     Longitude         0.000000
     Latitude          0.000000
     dtype: float64
```

```
[19]: features_one_hot = pd.get_dummies(
          features,
          columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial']
      )

      features_one_hot.head()
```

[19]:

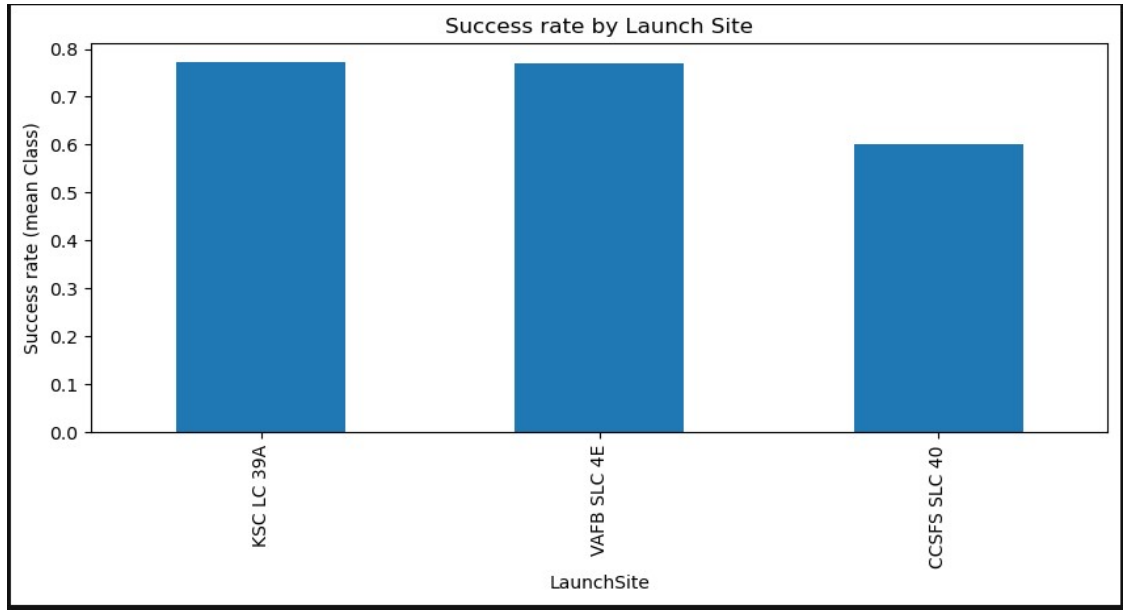
	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount
0	1	6123.547647	1	False	False	False	1.0	0
1	2	525.000000	1	False	False	False	1.0	0
2	3	677.000000	1	False	False	False	1.0	0
3	4	500.000000	1	False	False	False	1.0	0
4	5	3170.000000	1	False	False	False	1.0	0

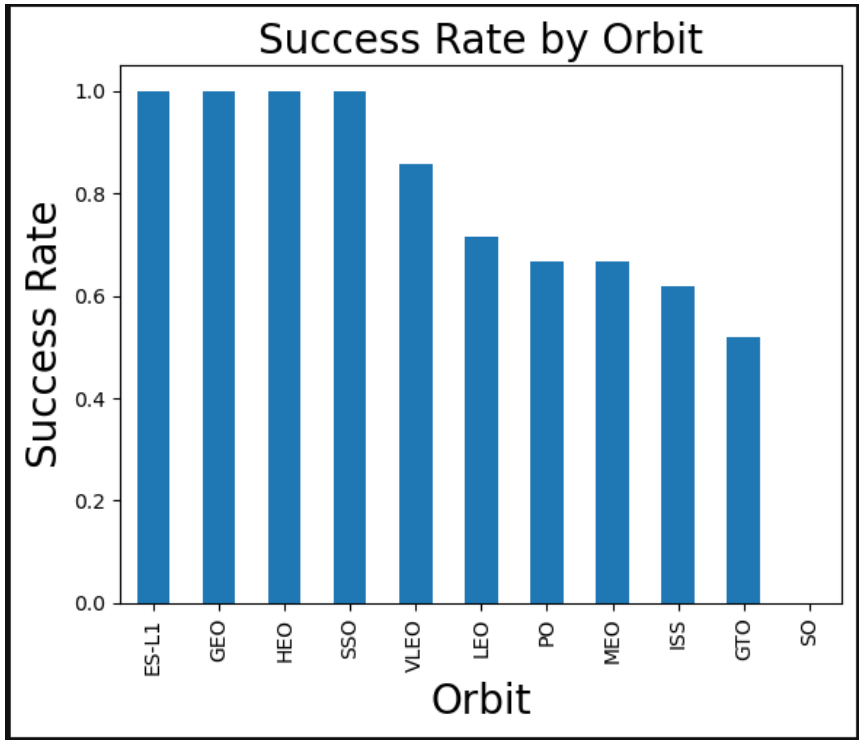
5 rows × 80 columns

EDA with Visualization

Launch site and orbit relationships

- Compared success patterns across LaunchSite and Orbit.
- Important note: GTO is a transfer orbit and should not be counted as GEO.
- Observed that some orbits/sites have higher success rates and different mission profiles.

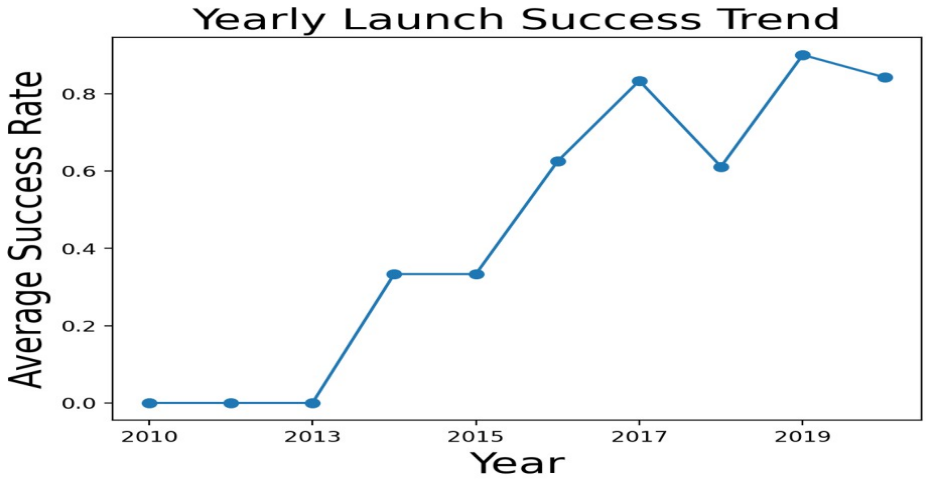


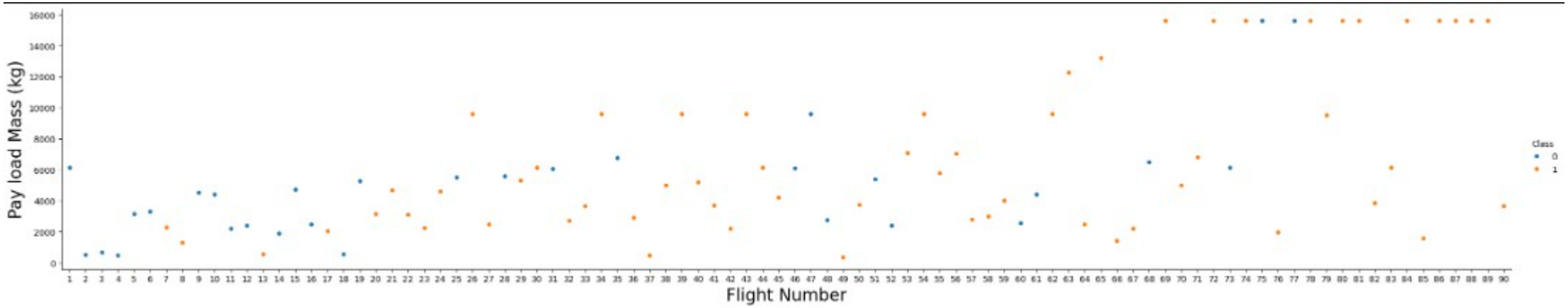


EDA with Visualization

Time trend and payload relationships

- Yearly trend: average success rate increases over time.
- FlightNumber vs PayloadMass: later flights show more successes even at higher payloads.
- Payload effect is not one-dimensional; it interacts with mission profile (orbit/site) and time.





EDA with SQL

SQLite + queries for structured insights

- Loaded Spacex.csv into SQLite (SPACEXTABLE).
- Executed queries to summarize launch sites, payload statistics, and landing outcomes.
- Used LIKE/TRIM to handle inconsistent text labels.

```
[13]: %%sql
SELECT *
FROM SPACEXTABLE
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5;
```

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

[13]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

```
[26]: %%sql
SELECT
  "Landing_Outcome",
  COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
```

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

[26]:

Landing_Outcome	outcome_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

```
[20]: %%sql
SELECT LOWER(TRIM(Mission_Outcome)) AS mission_outcome, COUNT(*) AS total
FROM SPACEXTABLE
GROUP BY LOWER(TRIM(Mission_Outcome))
ORDER BY total DESC;
```

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

[20]:

mission_outcome	total
success	99
success (payload status unclear)	1
failure (in flight)	1

EDA with SQL

Key query results (examples)

Example results from SPACEXTABLE (101 rows in the lab dataset):

Mission_Outcome counts (normalized):

Success (incl. variants) : 100

Failure (in flight) : 1

Other required SQL tasks:

```
[12]: %%sql
SELECT DISTINCT Launch_Site
FROM SPACEXTABLE
ORDER BY Launch_Site;
```

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

[12]:

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

```
[14]: %%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg
FROM SPACEXTABLE
WHERE Customer LIKE '%NASA (CRS)%';
```

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

[14]:

total_payload_mass_kg
48213

```
[16]: %%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass_kg
FROM SPACEXTABLE
WHERE Booster_Version = 'F9 v1.1'
  AND PAYLOAD_MASS__KG_ > 0;
```

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

[16]:

avg_payload_mass_kg
2928.4

```
[26]: %%sql
SELECT
  "Landing_Outcome",
  COUNT(*) AS outcome_count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
```

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

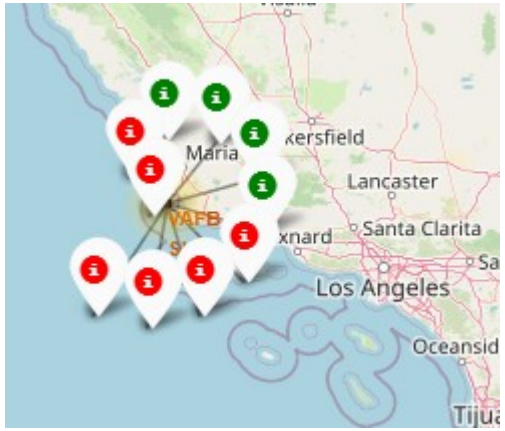
[26]:

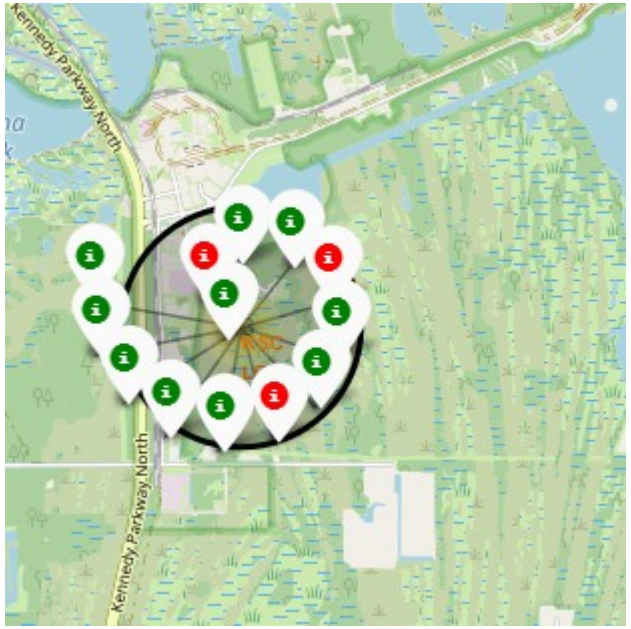
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

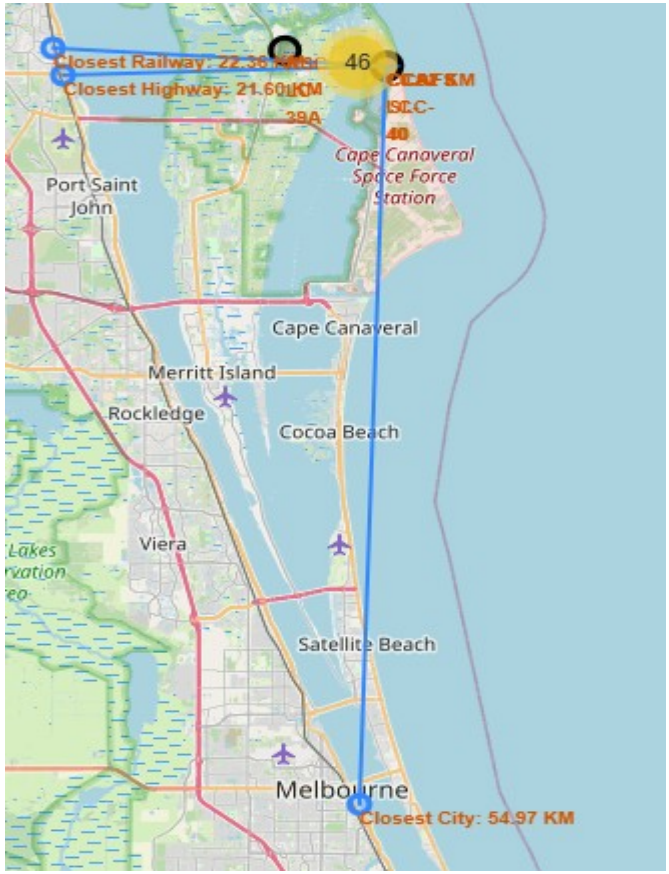
Interactive Visual Analytics

Folium launch site map & proximities

- Mapped launch sites and clustered launch outcomes (MarkerCluster).
- Green markers: success; red markers: failure.
- Calculated distances from launch site to closest city/railway/highway using Haversine distance.



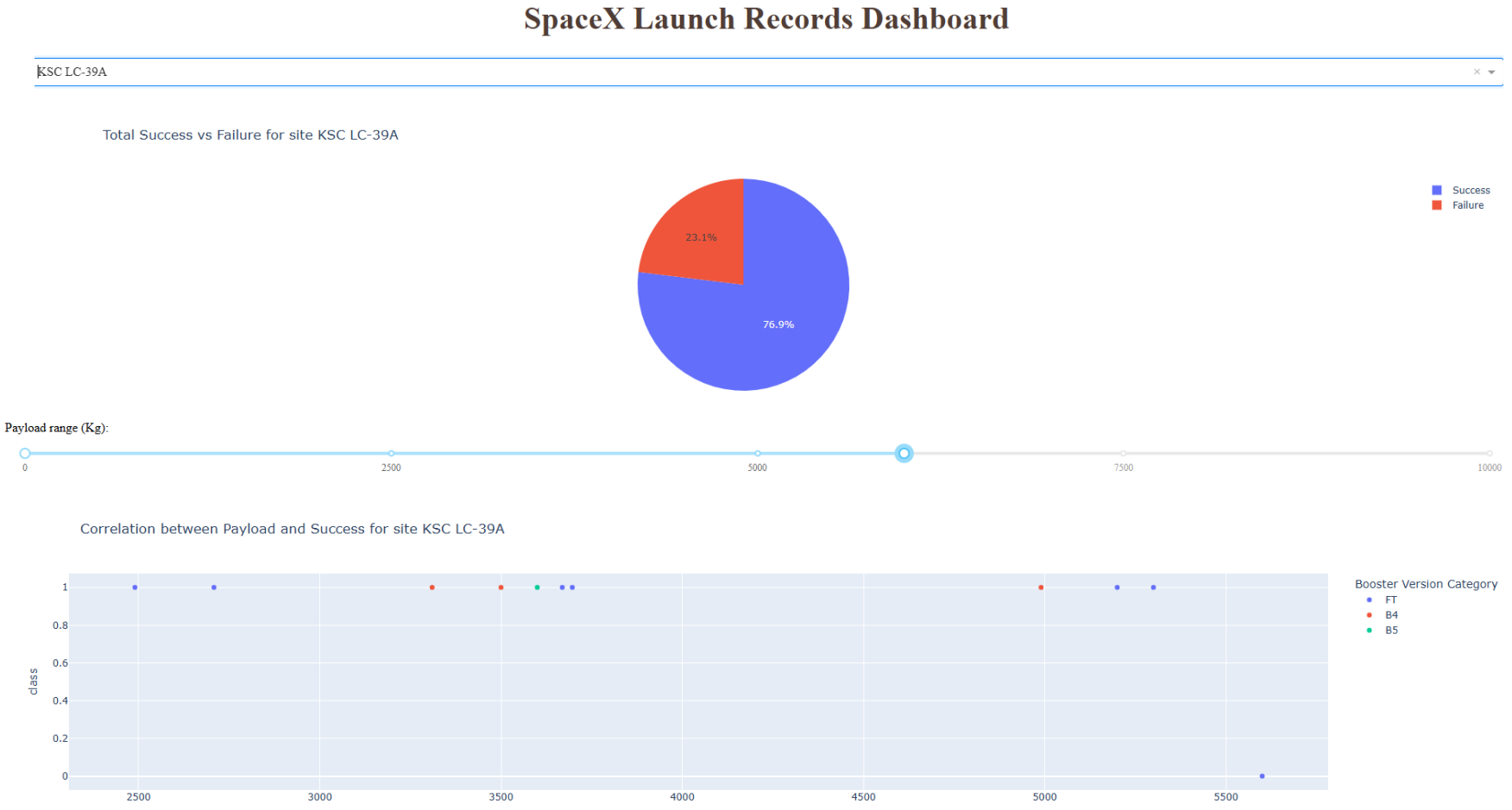




Interactive Dashboard

Plotly Dash app results

- Dropdown to select launch site (All Sites or specific site).
- Pie chart: total successes by site OR success vs failure for selected site.
- RangeSlider to filter payload mass; scatter plot shows class vs payload colored by booster category.



Predictive Analysis

Modeling workflow

- Prepared X from one-hot encoded features (dataset_part_4) and Y from Class (dataset_part_3).
- Standardized features (StandardScaler).
- Split data: train/test = 80/20 with random_state = 2.
- Hyperparameter tuning with GridSearchCV (cv = 10).
- Models: Logistic Regression, SVM, Decision Tree, KNN.

[10]:
```
transform = preprocessing.StandardScaler()
X = transform.fit_transform(X)
```

[11]:
```
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.2, random_state=2
)

print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```
```
(72, 80) (18, 80) (72,) (18,)
```

[18]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[19]:
```
svm_cv = GridSearchCV(svm, parameters, cv=10)

svm_cv.fit(X_train, Y_train)
```

[20]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```
```
tuned hpyerparameters :(best parameters)  {'C': np.float64(1.0), 'gamma': np.float64(0.03162277660168379), 'kernel': 'sigmoid'}
accuracy : 0.8482142857142858
```

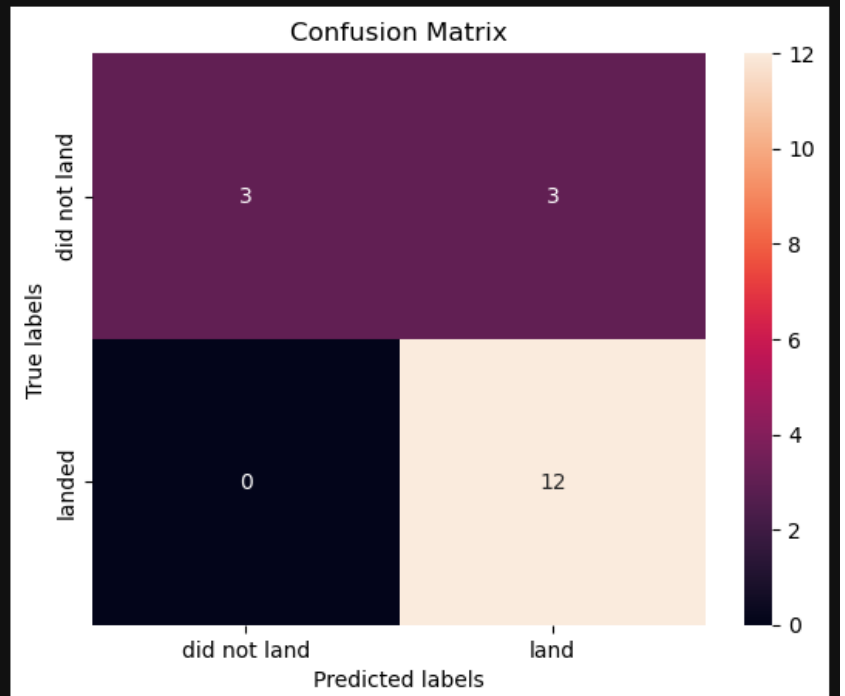
Predictive Analysis

Results and best model selection

Model comparison (example from run):

Model	Test accuracy	CV best_score	Notes
SVM	0.833	0.848	Winner (tie on test, better CV)
Logistic Regression	0.833	0.821	Very similar decision boundary
Decision Tree	0.667	0.888	Tuned via GridSearchCV
KNN	0.778	0.834	Tuned via GridSearchCV

```
[22]: yhat=svm_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```




```
[39]:
```

	Model	Best params	CV best_score	Test accuracy
0	SVM	{'C': 1.0, 'gamma': 0.03162277660168379, 'kern...	0.848214	0.833333
1	Logistic Regression	{'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'}	0.821429	0.833333
2	KNN	{'algorithm': 'auto', 'n_neighbors': 3, 'p': 1}	0.833929	0.777778
3	Decision Tree	{'criterion': 'gini', 'max_depth': 12, 'max_fe...	0.887500	0.666667

```
[43]: print('and the winner is:')
      print(results.loc[0, "Model"])

      and the winner is:
      SVM
```

Conclusion

What we learned & next steps

- Landing success improves over time, reflecting operational learning and hardware maturity.
- Orbit and launch site correlate with success rates (mission profile matters).
- Interactive tools (Folium, Dash) help communicate and explore outcomes quickly.
- SVM chosen as best-performing model under this dataset and evaluation setup.

Next steps (optional):

- Collect more recent launches and validate generalization.
- Try additional models (Random Forest, Gradient Boosting) and calibration.
- Improve feature set (weather, recovery ship availability, mission type details).