

# IBM Data Science Capstone

Safoora Naureen

# AGENDA

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

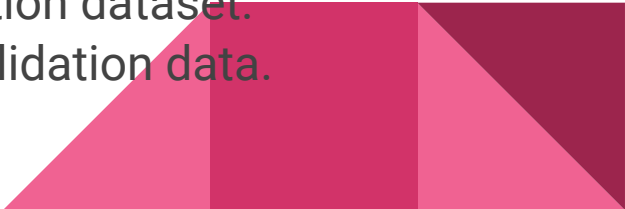


# EXECUTIVE SUMMARY

## SUMMARY Of METHODOLOGIES

- Data Collection.
- Data Wrangling.
- Exploratory Data Analysis with Visualization. Exploratory Data Analysis with SQL.
- Visual Analytics with Folium.
- Dashboard with Dash.
- Predictive Analysis working

## SUMMARY Of RESULT

- EDA Findings.
  - Prediction through Support vector machine, Classification Tree & Logistic.
  - For SVM sigmoid kernel provides better result on validation dataset.
  - Hyper parameters for decision tree classifier through validation data.
- 

# INTRODUCTION

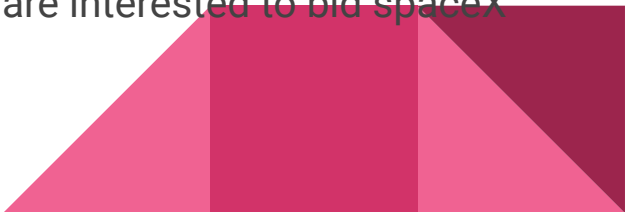
- Project background and context

- The spaceX advertises the launch of Falcon 9 rocket through their website, it has a cost of 62 million USD, whereas the other providers cost are 165 million USD each, the main reason of the

savings of spaceX is because that it can be reused the first stage.

- Problems you want to find answers

- We can determine the cost of the launch if we are determining that spaceX first stage will land.
  - The extracted information can be used for other companies who are interested to bid spaceX for launch of the rocket.



# METHODOLOGY

- Data Collection:
  - SpaceX Rest API.
  - Web Scrapping Wikipedia.
- Data Wrangling:
  - Hot encoding data fields and dropping irrelevant columns.
  - Exploratory Data Analysis (EDA) using visualization and SQL:
    - Scatter/bar graphs.
  - Interactive Visual Analytics using Folium and Plotly.
- Predictive Analysis using classification models:
  - Classification models.



## DATA COLLECTION VIA SPACEX API

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

# DATA COLLECTION VIA WEB SCRAPING

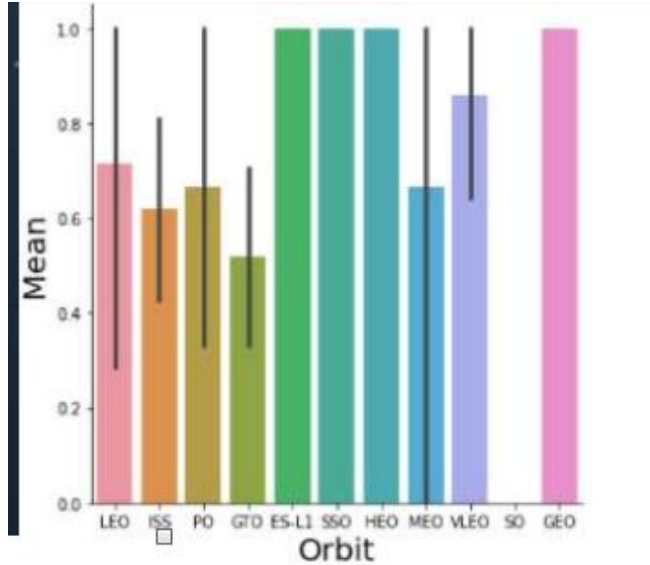
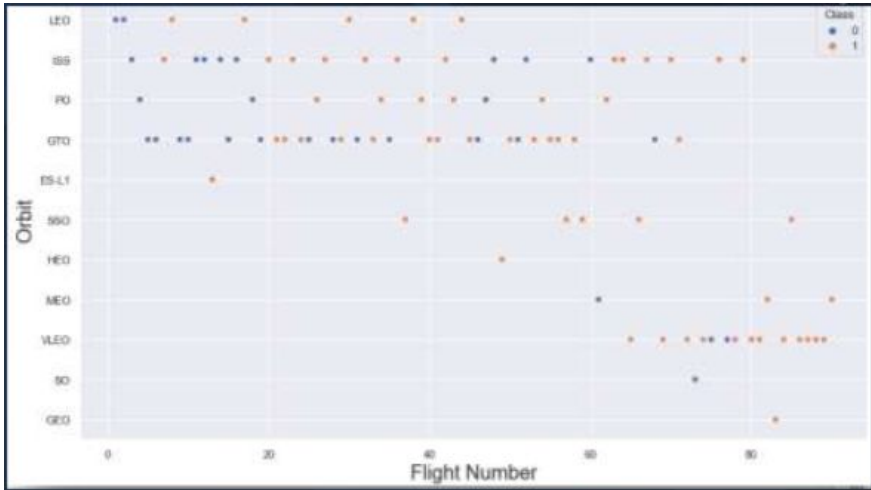
Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43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

# DATA WRANGLING

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004



# EDA with Data Visualisation



# EDA with SQL

```
In [6]: %sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
* ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

```
Out[6]:
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

# PREDICTIVE ANALYSIS

## Built the model

- Load the engineered data into a dataframe
- Transform and standardize the data using Numpy
- Split the data into training and test data sets
- Check how many samples were created and set our parameters/algorithms
- Fit the datasets into the GridSearchCV objects to train the model

## Evaluating the model

- Check the accuracy of the model and plot the Confusion Matrix
- Finding the best performing classification model
- Use the highest accuracy score



# KNN

```
: 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()  
  
: gscv = GridSearchCV(KNN,parameters,scoring='accuracy',cv=10)  
  knn_cv = gscv.fit(X_train,Y_train)  
  
: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)  
  print("accuracy :",knn_cv.best_score_)  
  
tuned hpyerparameters :(best parameters) {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}  
accuracy : 0.8482142857142858
```

## TASK 11

Calculate the accuracy of tree\_cv on the test data using the method score:

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

# Logistic regression

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

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

```
gscv = GridSearchCV(lr,parameters,scoring='accuracy',cv=10)  
logreg_cv = gscv.fit(X_train,Y_train)
```

We output the GridSearchCV object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

```
print("tuned hyperparameters :(best parameters) ",logreg_cv.best_params_)  
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hyperparameters :(best parameters) {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}  
accuracy : 0.8464285714285713
```

# SVM

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

```
gscv = GridSearchCV(svm,parameters,scoring='accuracy',cv=10)  
svm_cv = gscv.fit(X_train,Y_train)
```

```
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
```

# DECISION TREE

```
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()
```

```
gscv = GridSearchCV(tree,parameters,scoring='accuracy',cv=10)
tree_cv = gscv.fit(X_train,Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters) {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_s
amples_split': 2, 'splitter': 'best'}
accuracy : 0.8785714285714287
```

## TASK 9

Calculate the accuracy of tree\_cv on the test data using the method score:

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

```
accuracy: 0.9444444444444444
```

# Accuracy


(Decision Tree has THE BEST accuracy )

```
algo_df.rename(columns = {'index': 'Algorithm'}, inplace = True)  
algo_df.head()
```

	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.901786



# CONCLUSION

- . Number of launched success have increases over the years
  - ELS1, GEO, HEO, SSO have good success rate
  - KSCLC39 have best success rate.
  - Launch outcome impacted by the payload mass.
  - Accuracy is same on the test model
  - Decision tree classifier has highest accuracy.
- 

# APPENDIX

Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project.

