
IBM Developer
SKILLS NETWORK
Winning Space Race
with Data Science
Dpp du# didu
5 7wk Vhswhp ehu#5357

Rxwodqh

- H{hfxwlyh#Vxpp du|
- Lqwurgxfwlrq
- Phwkrgrorj|
- Uhvxow
- Frqfoxvlrq
- Dsshqgl{

H{hfxwlyh#Vxp p du|

- Vxp p du|#ri#p hwkrgrorj lhv
  - 0 Gdwd#Frohfwlrq#wkurxjk#DSL
  - 0 Gdwd#Frohfwlrq#z lwk#Z he#Vfudslqj
  - 0 Gdwd#Z udqjdqj
  - 0 H{sorudwru|#Gdwd#Dqdo|vlv#z lwk#VTO
  - 0 H{sorudwru|#Gdwd#Dqdo|vlv#z lwk#Gdwd#Ylvxdol}dwlrq
  - 0 Lqwhudfwlyh#Ylvxdo#Dqdo|wlfv#z lwk#Irdxp
  - 0 P dfklqh#Dhduqlqj#Suhglfwlrq
- Vxp p du|#ri#dq#hvxow
  - 0 H{sorudwru|#Gdwd#Dqdo|vlv#uhvxow
  - 0 Lqwhudfwlyh#dqdo|wlfv#lq#vfuhqvkrw
  - 0 Suhglfwlyh#Dqdo|wlfv#uhvxow

Lqwurgxfwlrq

- Surmhfw#edfnjurxqg#dqg#frqwh{w

  Vsdfh [dgyhuwlvhv Idofrq < urfnhw odxqfkhv rq lwv zhevlwh zlwk d frvw ri 95 p loolrq groodu> rwkhu surylghuv frvw xszdug ri 498 p loolrq groodu hdfk/ p xfk ri wkh vdylqjv lv ehfdxvh Vsdfh [fdq uhxvh wkh iluvw vwdjh1 Wkhuhiruh/ li zh fdq ghwhup lqh li wkh iluvw vwdjh z loo odqg/ zh fdq ghwhup lqh wkh frvw ri d odxqfk1 Wklv lqirup dwlrq fdq eh xvhg li dq dowhuqdwh frp sdq| zdqw wr elg djdlqvw vsdfh [iru d urfnhw odxqfk1 Wklv jrdo ri wkh surmhfw lv wr fuhdwh d p dfklqh ohduqlqj slshdqh wr suhglfw li wkh iluvw vwdjh z loo odqg vxffhvvixoo|1

- Sureohp v#|rx#z dqw#wr#ilqg#dqvz huv

  0 Z kdw#idfwruv#ghwhup lqh#li#wkh#urfnhw#z lo#odqg#vxffhvvixoo|B

  0 Wkh#lqwhudfwlrq#dp rqjvw#ydulrxv#ihdwxuhv#wkdw#ghwhup lqh#wkh#vxffhvv#udwh#ri#d#vxffhvvixo# odqglqj1

  0 Z kdw#rshudwlqj#frqglwlrqv#qhhgv#wr#eh#lq#sodfh#wr#hqvxuh#d#vxffhvvixo#odqglqj#surjudp 1

Section 1

Methodology

Phwkrgrorj|

H{hfxwlyh#Vxpp du|

- Gdwd#frohfwlrq#p hwkrgrorj|=
  - Gdwd#zdv#frohfwhg#xvlqj#Vsdfh[#DSI#dqg#zhe#vfudslqj#urp#Z lnlshgld1
- Shuirup#gdwd zudqjdqj
  - Rqh0krw#hqfrglqj#zdv#dssdhg#wr#fdwhjrulfdd#hdwxuhv
- Shuirup#h{sorudwru|#gdwd#dqdd|vlv#HGD,#xvlqj#ylvxddl}dwlrq#dqg VTO
- Shuirup#lqwhudfwlyh#ylvxdd#dqdd|wlfv#xvlqj#Iroxp#dqg Sorwo| Gdvk
- Shuirup#suhglfwlyh#dqdd|vlv#xvlqj#fodvvlilfdwlrq#p rghov
  - Krz#wr#exlog/#wxqh/#hydoxdwh#fodvvlilfdwlrq#p rghov

Gdwd#Frohfwlrq

- Wkh gdwd z dv frohfwhg xvlqj ydulrxv p hwkrgv
  - 0 Gdwd frohfwlrq z dv grqh xvlqj jhwuhtxhvwwr wkh Vsdfh[DSL1
  - 0 Qh{w/ z h ghfrghg wkh uhvsrqvh frqwhqw dv d Mvrq xvlqj 1mvrq+, ixqfwlrq fdoo dqg wxuq lw lqwr d sdqgdv gdwdiudp h xvlqj 1mvrqbqrup dd}h+,1
  - 0 Z h wkhq fdoodqhg wkh gdwd/ fkhfnhg iru p lvvlqj ydoxhv dqg ilo lq p lvvlqj ydoxhv z khuh qhfhvvdu|1
  - 0 Lq dgglwlrq/ z h shuirup hg z he vfudslqj urp Z lnlshgld iru Idcfrq < oxqfk uhfrugv z lwk Ehdxwlixo Vrxs1
  - 0 Wkh remfwlyh z dv wr h{wudfw wkh oxqfk uhfrugv dv KWP O wdedn/ sduvh wkh wdedn dqg frqyhuw lwwr d sdqgdv gdwdiudp h iru ixwxuh dqdo|vlv1

Gdwd#Froohfwlrq#D Vsdfh[#DSL

- Z h#xvhg#wkh#jhw#uhtxhvw#wr#wkh# Vsdfh[#DSI#wr#froohfw#gdwd/#fohdq# wkh#uhtxhvwhg#gdwd#dqg#glg#vrp h# edvlf#gdwd#z udqjdqj#dqg# irup dwwlqj1

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()

In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Gdwd#Froohfwlrq#0 Vfudslqj

- Z h#dssolhg#z he#vfudsslqj#wr# z hevfuds#Iddfrq#k #odxqfk#uhfrugv# z lwk#Ehdxwlixo Vrxs#
- Z h#sduvhg#wkh#wdeoh#dqg#frqyhuwhg# lw#lqwr#d#sdqgdv#gdwdiudph

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        html_data = requests.get(static_url)
        html_data.status_code
```

```
Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html_data.text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
In [7]: # Use soup.title attribute
        soup.title
```

```
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

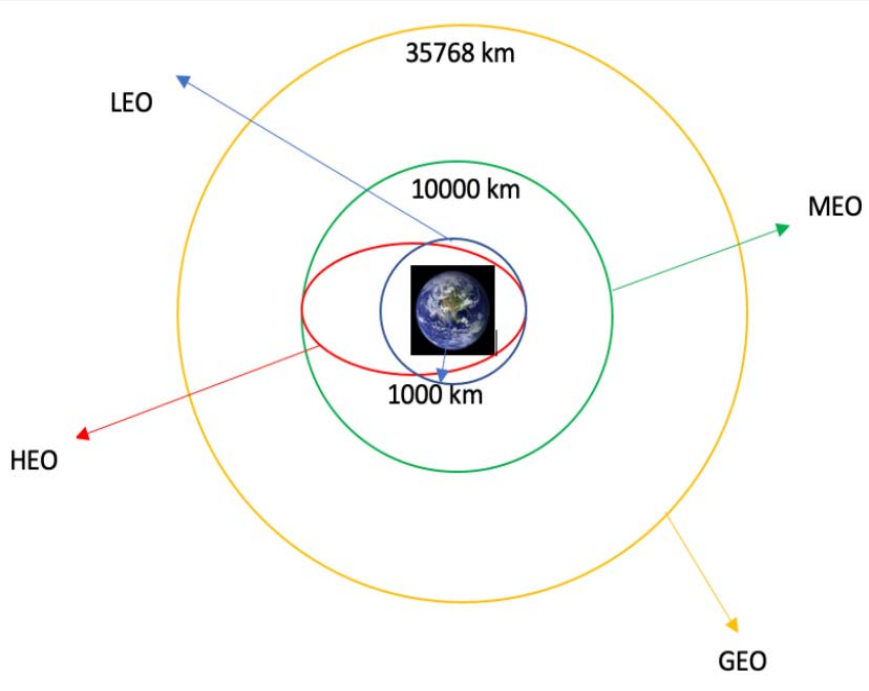
3. Extract all column names from the HTML table header

```
In [10]: 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

         element = soup.find_all('th')
         for row in range(len(element)):
             try:
                 name = extract_column_from_header(element[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

<

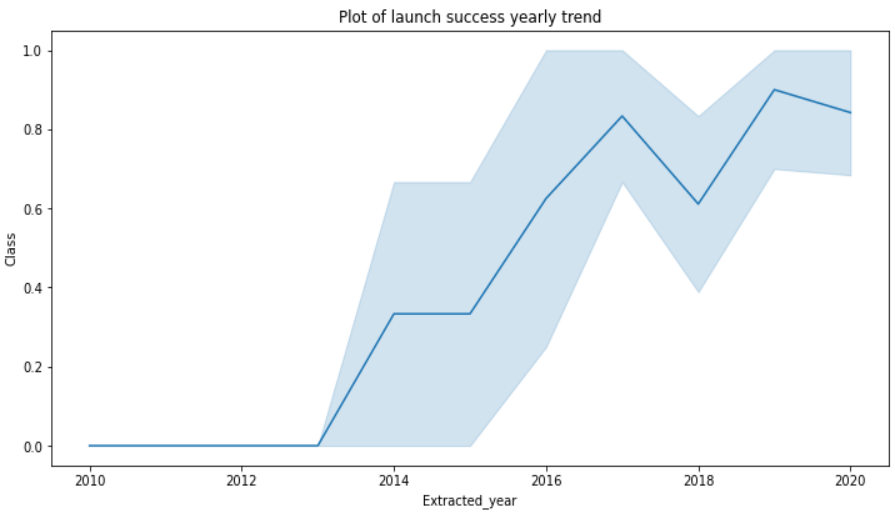
G dwd#Z udqj dqj



- Z h#shuirup hg#h{sorudwru|#gdwd#dqdo|vlv# dqg#ghwhup lqhg#wkh#wudlqlqj#odehov1
- Z h#fdofxodwhg#wkh#pxp ehu#ri#odxqfkhv#dw# hdfk#vlwh/#dqg#wkh#pxp ehu#dqg#rffxuhqfh# ri#hdfk#rue lw
- Z h#fuhdwhg#dqg lqj#rxwfrp h#odehd#urp # rxwfrp h#froxp q#dqg#h{sruwhg#wkh#hvxow# wr#fvy1

HGD#Z lwk#Gdwd#Ylvxdd}dwlrq

- Z h#h{soruhg#wkh#gdwd#e|#ylvxdd}lqj# wkh#uhodwlrqvkls#ehwz hhq#idjkw# qxp ehu#dqg#odxqfk#Vlwh/#sd|orgg#dqg# odxqfk#vlwh/#vxffhvv#udwh#r#hdfk#ruelw# w|sh/#idjkw#qxp ehu#dqg#ruelw#w|sh/#wkh# odxqfk#vxffhvv#|hduv|#wuhqg1#



HGD#z lwk VTO

- Z h#orghg#wkh#Vsdfh[#gdwdvhw#lqwr#d#SrvwjuhVTO#gdwdedvh#z lwkrxw#ohdylqj# wkh#mxs|whuqrwherrn1
- Z h#dssolhg#HGD#z lwk#VTO#wr#jhw#lqvljkw#iurp #wkh#gdwd1#Z h#z urwh#txhulhv#wr# ilqg#rxw#iru#lqvwdqfh=
    - 0 Wkh#qdp hv#ri#xqltxh#odxqfk#vlwhv#lq#wkh#vsdfh#p lvvlrq1
    - 0 Wkh#wrwdo#sd|ordg#p dvv#fduulhg#e|#errvwhuv#odxqfkhg#e|#QDVD#+FUV,
    - 0 Wkh#dyhudjh#sd|ordg#p dvv#fduulhg#e|#errvwhu#yhuvlrq#I<#y414
    - 0 Wkh#wrwdo#qxp ehu#ri#vxffhvvixo#dqg#idloxuh#p lvvlrq#rxwfrp hv
    - 0 Wkh#idlohg#odqglqj#rxwfrp hv#lq#gurqh#vkls/#wkhlu#errvwhu#yhuvlrq#dqg#odxqfk#vlwh#qdp hv1

Exlog#dq#Lqwhudfwlyh#P ds#Z lwk#Irdxp

- Z h#p dunhg#do#oxqfk#vlwhv/#dqg#dgghg#p ds#remfw#vxfk#dv#p dunhuv/#flufohv/#dqhv#wr# p dun#wkh#vxffhvv#ru#idloxuh#ri#oxqfkhv#iru#hdfk#vlwh#rq#wkh#irdxp #p ds1
- Z h#dvvljqhg#wkh#hdwxuh#oxqfk#rxwfrp hv#idloxuh#ru#vxffhvv,#wr#fodvv#3#dqg#4 111/#3# iru#idloxuh/#dqg#4#iru#vxffhvv1
- Xvlqj#wkh#froruCodehdg#p dunhu#fxvwhuv/#z h#bghqwlilhg#z klfk#oxqfk#vlwhv#kdyh# uhodwlyhq|#klj k#vxffhvv#udwh1#
- Z h#fdofxodwhg#wkh#glvwdqfhv#ehwz hhq#d#oxqfk#vlwh#wr#lw#sur{lp lwhv1#Z h#dqvz huhg# vrp h#txhvwlrq#iru#bqvwdqfh=

  0 Duh#oxqfk#vlwhv#qhdu#udlz d|v/#klj kz d|v#dqg#frdvwdqhv1

  0 Gr#oxqfk#vlwhv#nhhs#fhuwdlq#glvwdqfh#dz d|#urp #flwlhv1

Exlog#d#Gdvkerdug#zlwk#Sorwo|#Gdvk

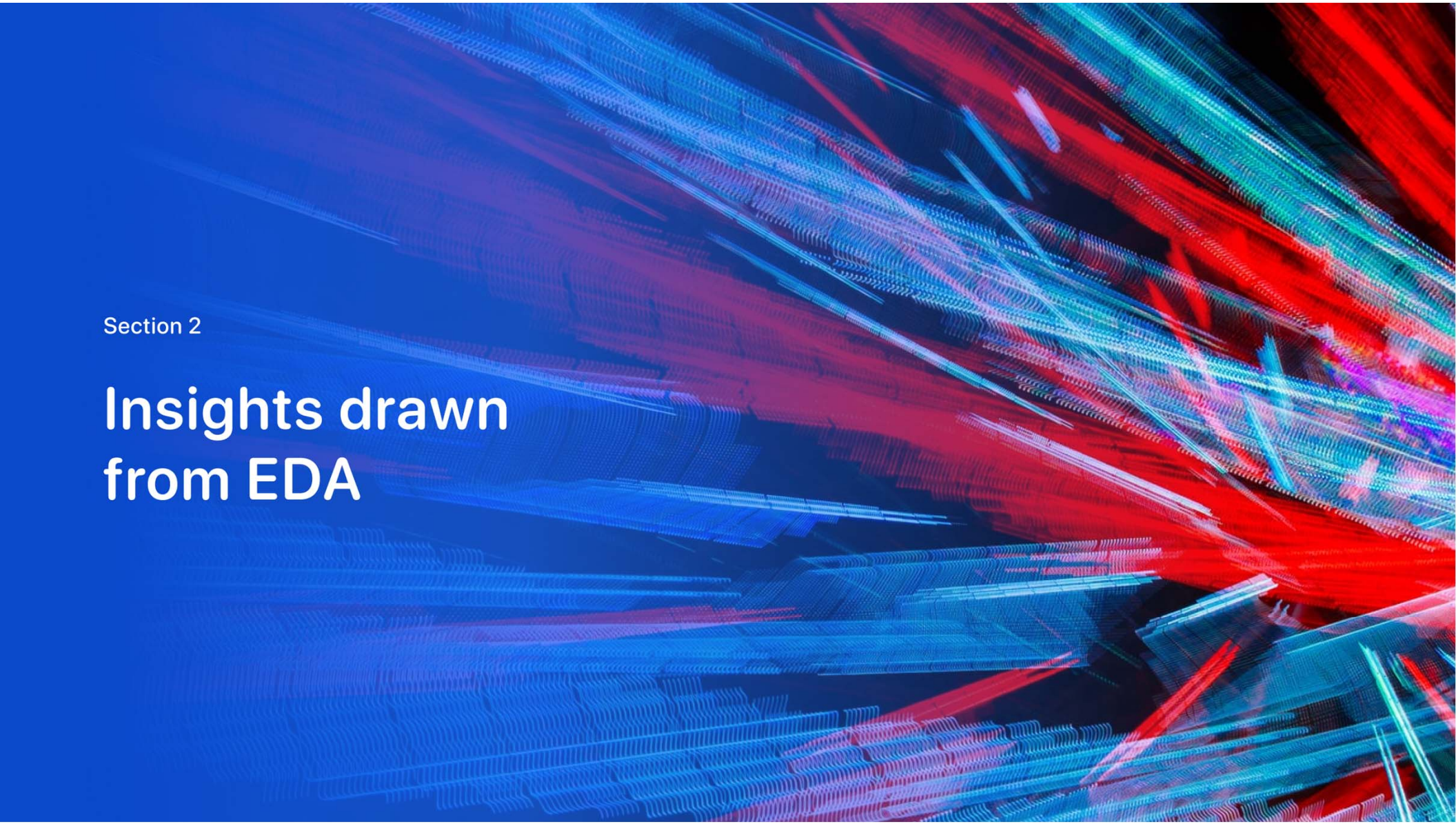
- Zh#exlow#dq#lqwhudfwlyh#gdvkerdug#zlwk#Sorwo|#gdvk
- Zh#sorwwhg#slh#fkduw#vkrzlqj#wkh#wrwdo#oxqfkhv#e|#d#fhuwdlq#vlwhv
- Zh#sorwwhg#vfdwwhu#judsk#vkrzlqj#wkh#hodwlrqvkls#zlwk#Rxwfrph#dqg#Sd|ordg#Pdvv#Nj,#iru#wkh#gliihuhqw#errvwhu#yhuvlrq

Suhglfwlyh#Dqdo|vlv#Fodvvlilfdwlrq,

- Z h#rdghg#wkh#gdwd#xvlqj#qxp s| dqg#sdqgdv/#wudqvirup hg#wkh#gdwd/#vr#w#rxu# gdwd#lqwr#wudlqlqj#dqg#whvwlqj1
- Z h#exlw#gliihuhqw#p dfklqh#hduqlqj#p rgho#dqg#wxqh#gliihuhqw# k|shusdudp hwhuv#xvlqj#JulgVhdufkFY1
- Z h#xvhg#dffxudf|#dv#wkh#p hwulf#ru#rxu#p rgho/#p suryhg#wkh#p rgho#xvlqj# ihdwxuh#hqjlqhhulqj#dqg#dojrulwkp #wxqlqj1
- Z h#irxqg#wkh#ehvw#shuirup lqj#fodvvlilfdwlrq#p rgho1

Uhvxow

- H{sorudwru|#gdwd#dqdo|vlv#uhvxow
- Lqwhudfwlyh#dqdo|wlfv#ghp r#lq#vfuhqvkrw
- Suhglfwlyh#dqdo|vlv#uhvxow

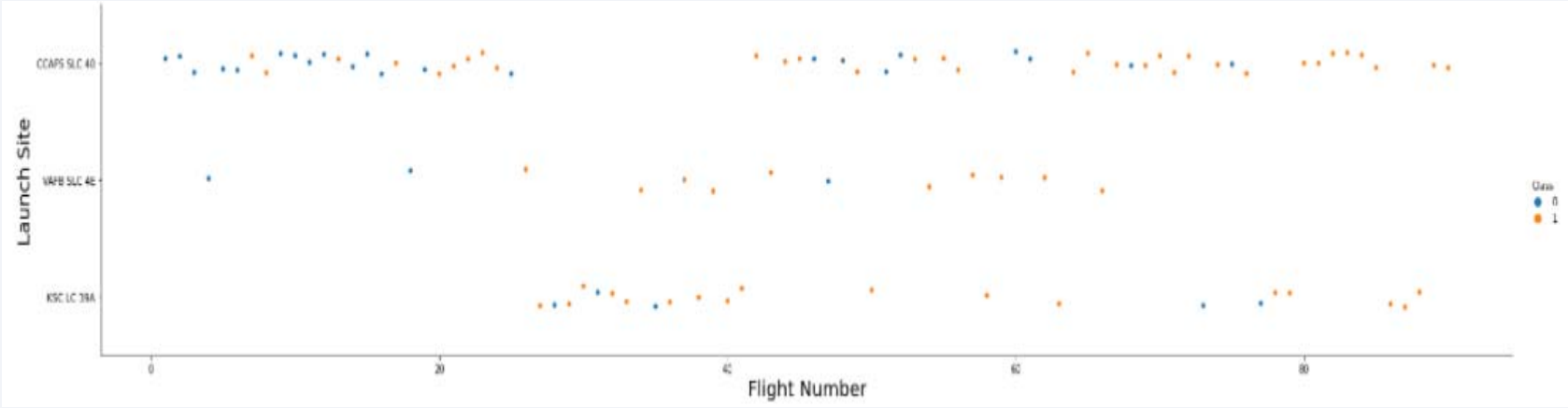


Section 2

Insights drawn from EDA

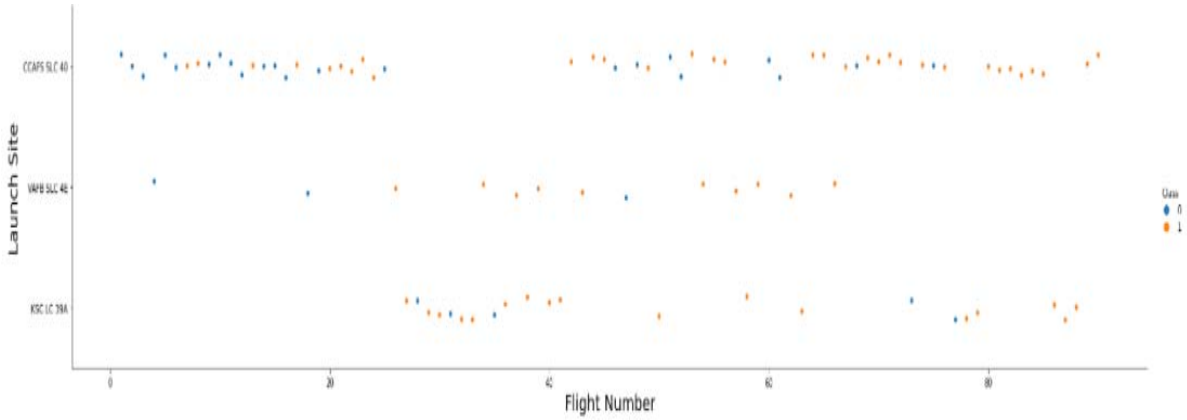
Idjkw#Qxp ehu#yv1#Odxqfk Vlwh

- Iurp #wkh#sorw/#z h#irxqg#wkdw#wkh#oujhu#wkh#idjkw#p rxqw#dw#d#odxqfk#vlwh/#wkh#juhdwhu# wkh#vxffhvv#udwh#dw#d#odxqfk#vlwh1



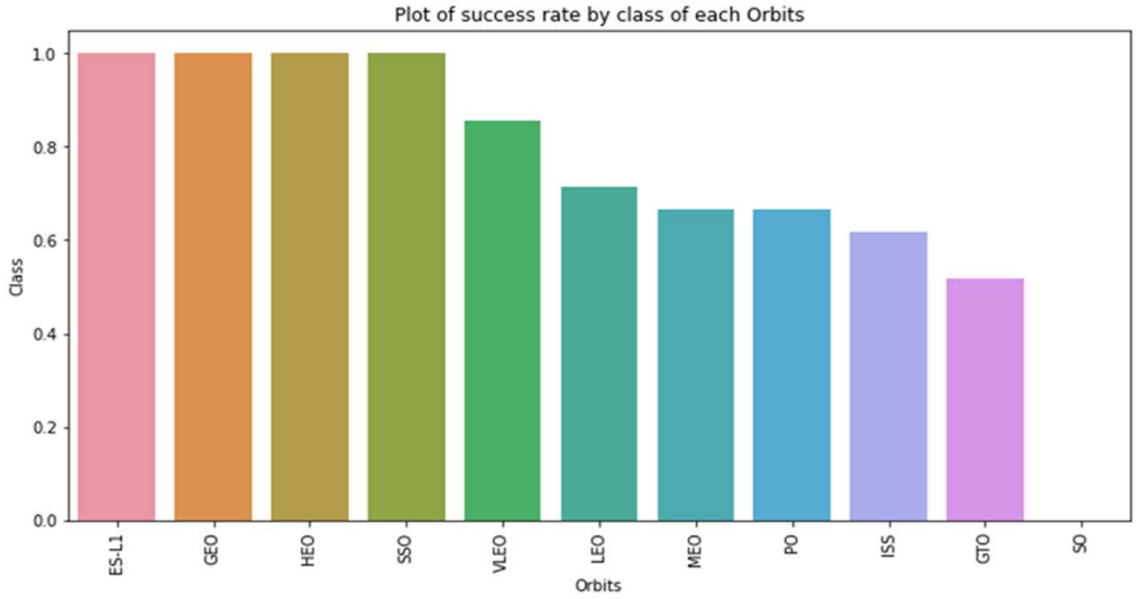
Sd|ordg#yv1# Odxqfk#Vlwh

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



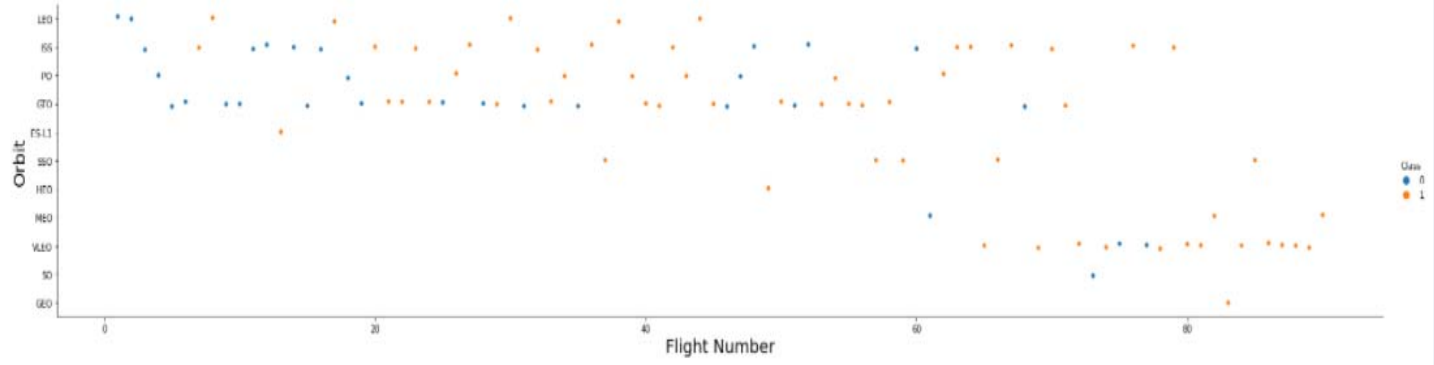
Vxffhvv#Udwh#yv1#Ruelw#W|sh

- Iurp #wkh#scrw#z h#fdq#vhh# wkdw#HV04 /#JHR /#KHR /#VVR /# YOHR #kdg#wkh#p rvw#vxffhvv# udwh1



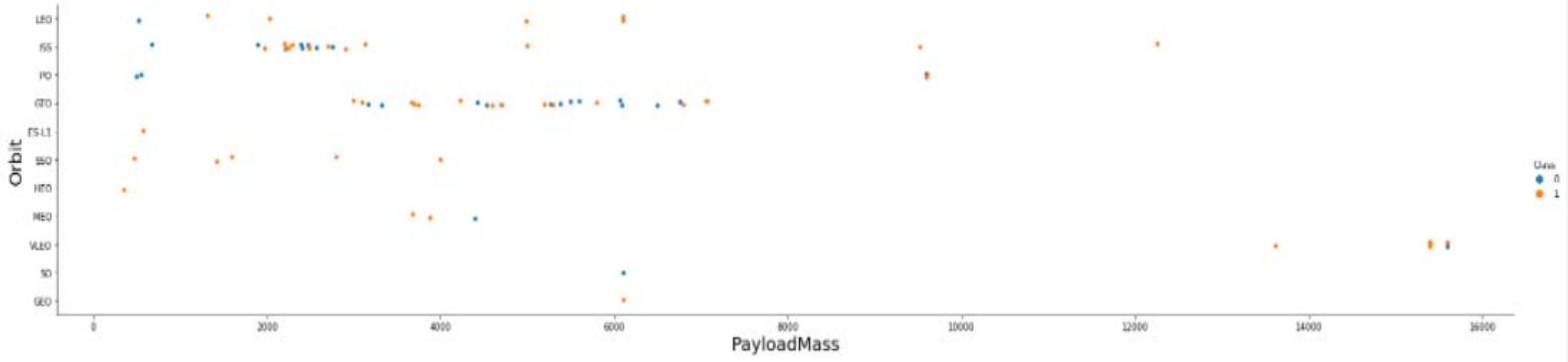
Idjkw#Qxp ehu#yv1#Ruelw#W|sh

- Wkh#sorw#ehorz#vkrzv#wkh#Idjkw#Qxp ehu#yv1#Ruelw#w|sh1#Z h#revhuyh#wkdw#lq#wkh#DHR # ruelw#vxffhvv#lv#lqfuhdvlqj#ryhu#wkh#qxp ehu#ri#idjkw#z khuhdv#lq#wkh#JWR #ruelw#wkhuh#lv# qr#ulvdwlrqvkls#ehwz hhq#idjkw#qxp ehu#dqg#wkh#ruelw1#



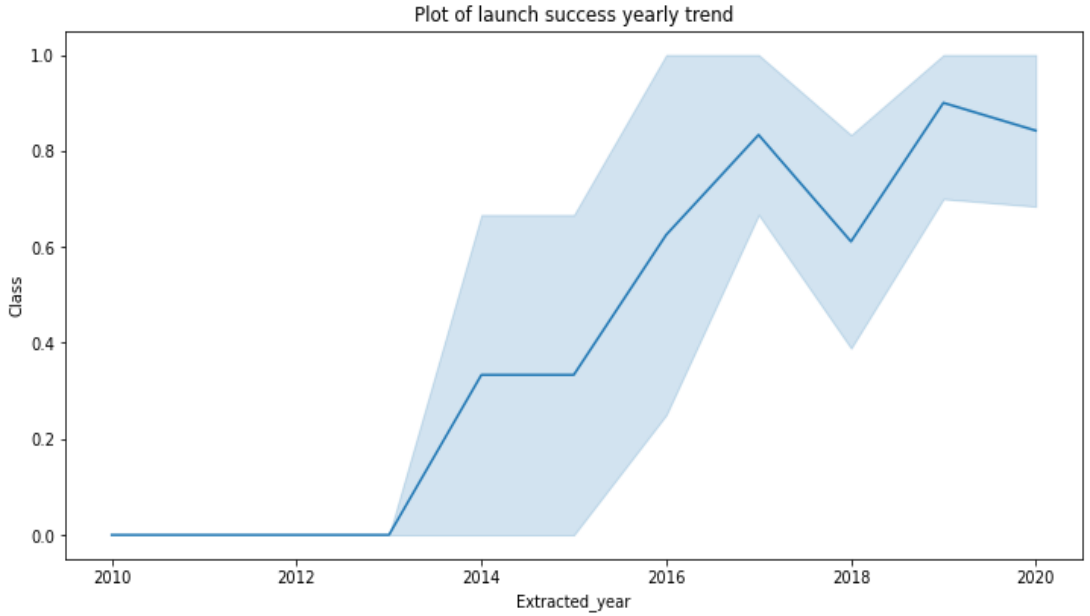
Sd|ordg#yv1#RuelwW|sh

- Z h#fdq#revhuyh#wkdw#z lk#khdy|#sd|ordgv/#wkh#vxffhvvixo#odqglqj#duh#p ruh#iru#SR /# OHR #dqg#LVV#ruelw1



Launch Success Yearly Trend

- From the plot, we can observe that the success rate since 2013 kept increasing till 2020



Display the names of the unique launch sites in the space mission

- Write a query to find the unique launch sites from the SpaceX data.

Display the names of the unique launch sites in the space mission

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Odxqfk#Vlwh#Qdp hv#Ehjlq#z lwk#FFD*

Display 5 records where launch sites begin with the string 'CCA'

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Z h#xvhg#wkh#txhu|#deryh#wr#glvsod|#8#uhfrugv#z khuh#odxqfk#vlwhv#ehjlq#z lwk#cFFDc

Wrwdo#Sd|ordg#P dvv

- Z h#fdofxodwhg#wkh#wrwdo#sd|ordg#fduulhg#e|#errvwhuv#urp #QDVD#dv#788<9# xvlqj#wkh#txhu|#ehorz

Display the total payload mass carried by boosters launched by NASA (CRS)

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Dyhudjh# Sd|ordg#P dvv# e|#I<#y414

- Z h#fdofxodwhg#wkh#dyhudjh# sd|ordg#p dvv#fduulhg#e|# errvwhu#yhuvlrq#I<#y414#dv# 5<5;17

Display average payload mass carried by booster version F9 v1.1

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

I luv#Vxffhvvixo#Jurxqg#Odqglqj#Gdwh

- Z h#revhuyhg#kdw#wkh#gdwhv#ri#wkh# iluw#vxffhvvixo#odqglqj#rxwfrp h#rq# jurxqg#sdg#z dv#55qg Ghfhp ehu# 5348

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Vxffhvvixo#Gurqh#Vkls#Ddqglqj#z lwk#Sd|ordg#ehwz hhq# 7333#dqg#9333

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- Z h#xvhg#wkh#**WHUH** fodxvh#wr# ilowhu#ru#errvwhuv#z klfk#kdyh# vxffhvvixo|#odqghg#rq#gurqh# vkls#dqg#dssdhg#wkh#**DQG** frqglwrq#wr#ghwhup lqh# vxffhvvix#odqglqj#z lwk#sd|ordg# p dvv#juhdwhu#wkdq#7333#exw#bhvv# wkdq#9333

Wrwdo#Qxp ehu#ri#Vxffhvvixo#dqg#Idloxuh P lvvlrq# Rxwfrp hv

List the total number of successful and failure mission outcomes

In [16]:

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

Out[16]:

	failureoutcome
0	1

- Z h#xvhg#z lqgfdug#dnh#p(Ã#wr# ilqgu#ru#Z KHUH P lvvlrqRxwfrp h z dv#d#vxffhvv#ru#d#idloxuh1#

Errvwhuv#Fduulhg# Pd{lpxp Sd|ordg

- Z h#ghwhup lqhg#wkh#errvwhu#wkdw# kdyh#fduulhg#wkh#pd{lpxp # sd|ordg#kvlqj#d#vxevxhu|#lq#wkh# **Z KHUH** fodxvh#dqg#wkh#**PD[+#** ixqfwlrq1

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

5348#Qdxqfk#Uhfrugv

- Z h#xvhg#d#frp elqdwrqv#r i#wkh#Z KHUH fodxvh/#DQG/#dqg#EHWZ HHQ frqglwrqv#wr#ilowhu#ru#idlong odqglqj#rxwfrp hv#q#gurqh#vkls/#wkhlu#errvwhu# yhuvlrqv/#dqg odxqfk#vlwh#qdp hv#ru#|hdu#5348

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Udqn#Ddqglqj#Rxwfrp hv#Ehwz hhq#53430390 37#dqg#534:03605 3

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- Z h#vhohfwhg#Ddqglqj#rxwfrp hv# dqg#wkh#**FRXQW** ri#olqglqj# rxwfrp hv#urp #wkh#gdwd#dqg# xvhg#wkh#**Z KHUH** fodxvh#wr#ilowhu# iru#olqglqj#rxwfrp hv#**EHWZ HHQ** 53430390 37#wr#53430360531
- Z h#dssolhg#wkh#**JURXS#E\#**fodxvh# wr#jurxs#wkh#olqglqj#rxwfrp hv# dqg#wkh#**RUGHU#E\#**fodxvh#wr# rughu#wkh#jurxsh g#olqglqj# rxwfrp h#q#ghvfhqglqj#rughu1

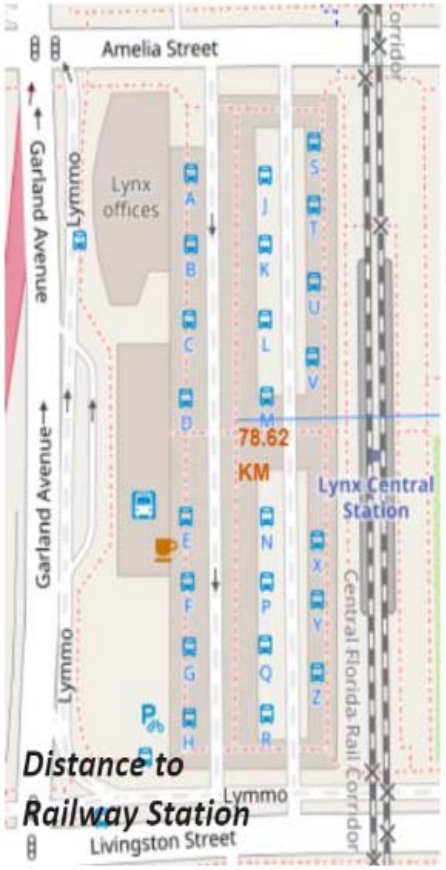
Section 4

Launch Sites Proximities Analysis

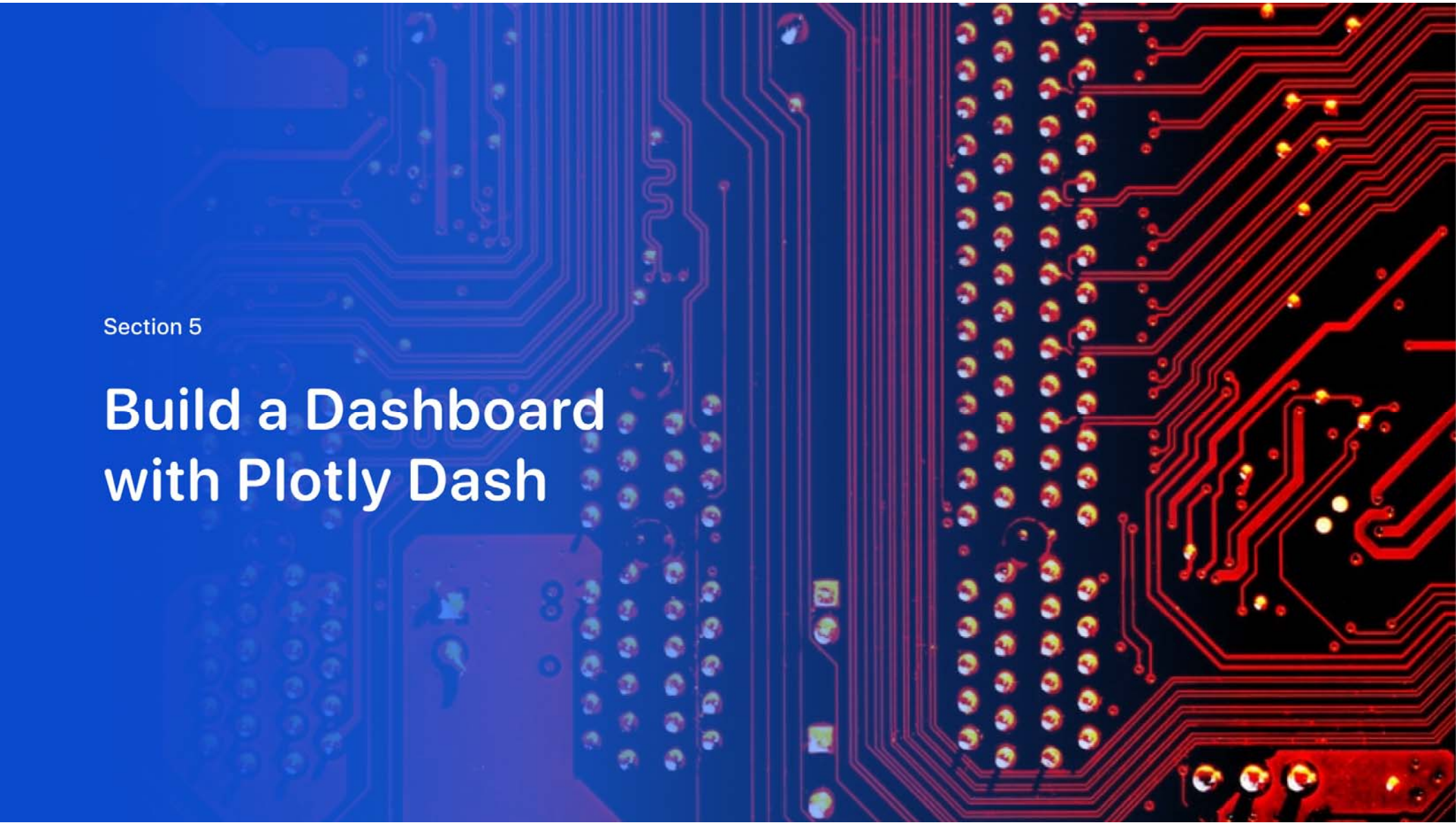
Doo#odxqfk#vlwhv#joredo#pds#pdunhuv

Launch sites locations on the map

Odxqfk#Vlwh#glvwdqfh#wr#odqgp dunv



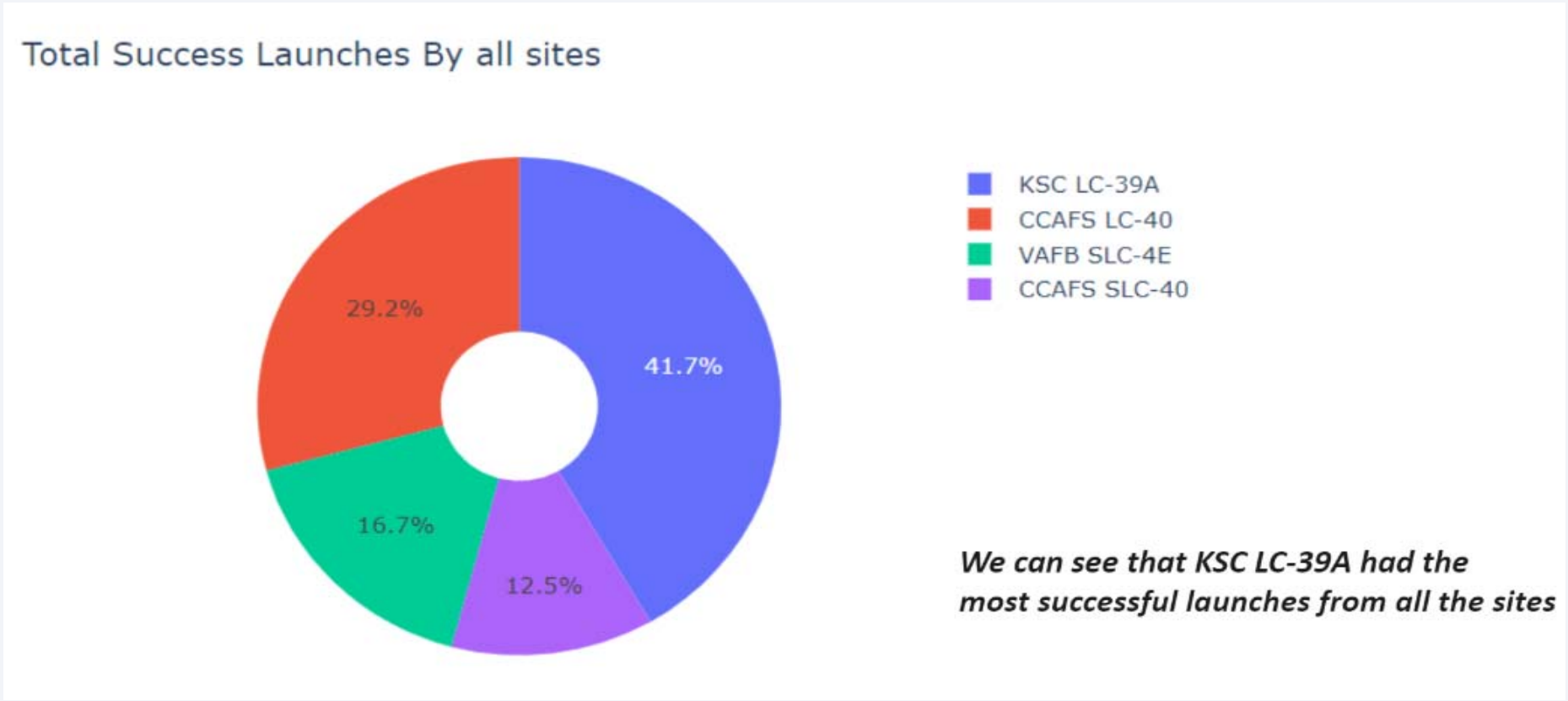
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



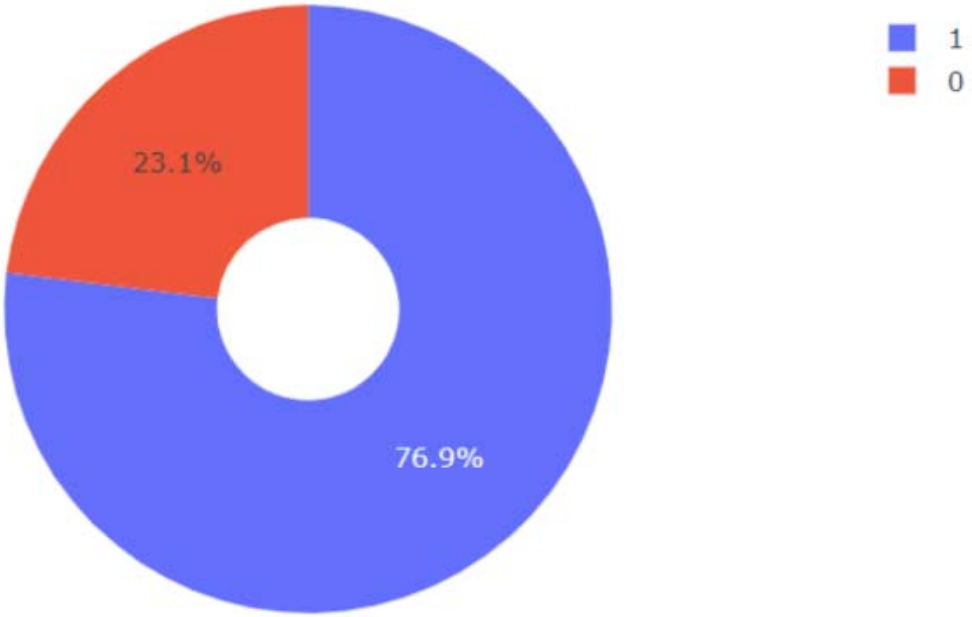
Section 5

Build a Dashboard with Plotly Dash

Slh#fkduw#vkrz lqj#wkh#vxffhvv#shufhqwdjh#dfklhyhg#e|#hdfk#odxqfk#vlwh

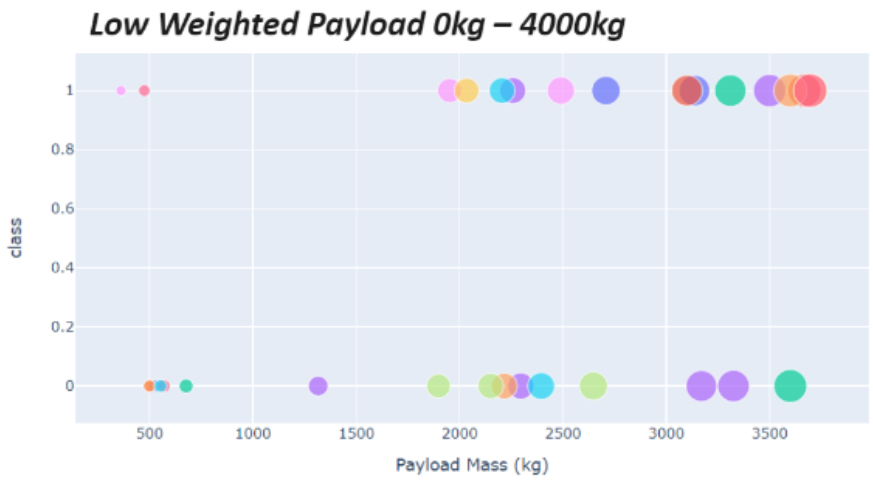


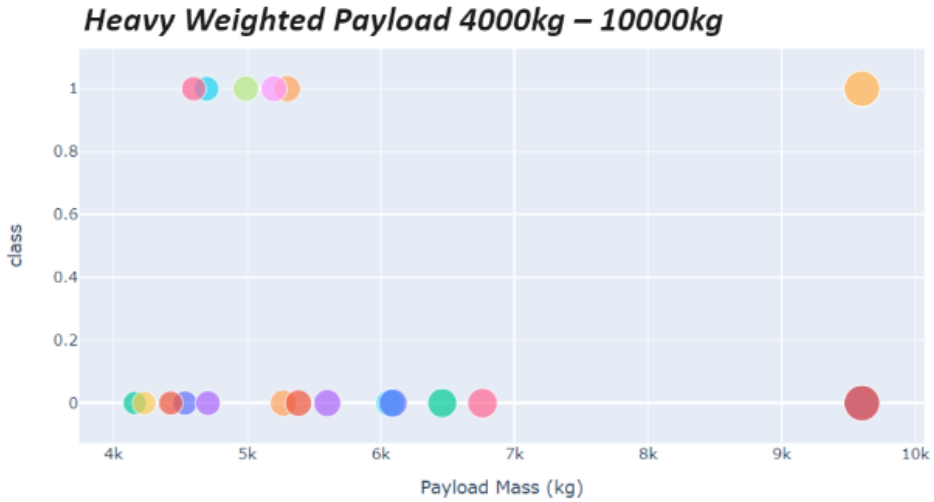
Sh#fkdw#vkrz lqj#wkh#Ddxqfk#vlwh#z lwk#wkh#kljkhvw#odxqfk#vxffhvv#udwh



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Vfdwwhu#sorw#ri#Sd|ordg#ty v#Ddxqfk#Rxwfrp h#iru#doo#vlwhv/#z lwk#gliihuhqw# sd|ordg#vhohfwhg#lq#wkh#udqj h#vdghu





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

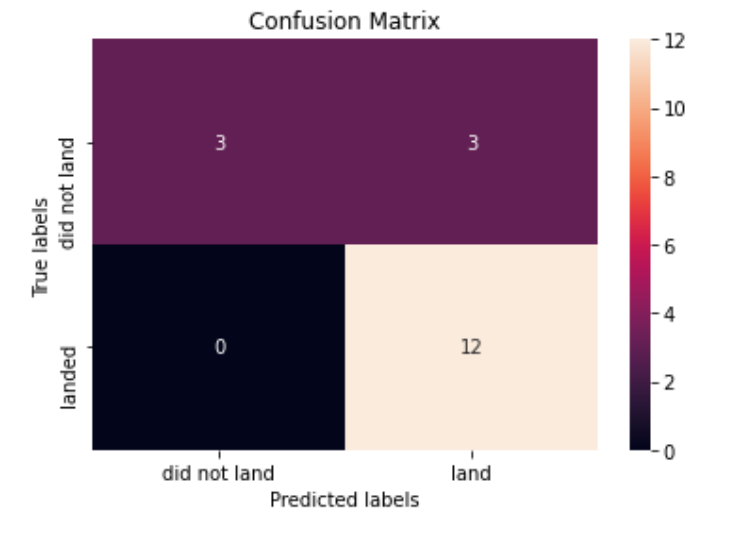
- The decision tree classifier is the model with the highest classification accuracy

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Frqixvlrq#P dwul{

- Wkh#frqixvlrq#p dwul{#ru#wkh#ghflvlrq#wuhh# fodvvlilhu#vkrz v#wkdw#wkh#fodvvlilhu#fdq# glvwlqjxlvk#ehwz hhq#wkh#gliihuhqw#fodvvhv1# Wkh#p dmru#sureolp #lv#wkh#idoh#srvlwlyhv#l1h1/# xqvxffhvvixo#olqg lqj#p dunhg#dv#vxffhvvixo# olqg lqj#e|#wkh#fodvvlilhu1



Frqfoxvlrqv

Z h#fdq#frqfoxgh#wkdw=

- Wkh#oduj hu#wkh#idj kw#bp rxqw#bw#d#olxqfk#vlwh/#wkh#j uhdwhu#wkh#vxffhvv#udwh#bw#d#olxqfk#vlwh1
- Olxqfk#vxffhvv#udwh#vwduwhg#wr#lqfuhdvh#lq#5346#wlo#5351
- Ruelw#HV04/#JHR/#KHR/#VVR/#YOHR#kdg#wkh#p rvw#vxffhvv#udwh1
- NVF#OF06<D#kdg#wkh#p rvw#vxffhvvixo#olxqfkhv#ri#dq|#vlwhv1
- Wkh#Ghflvlrq#wuhh#fodvvlilhu#v#wkh#ehvw#p dfklqh#bnduqlqj#dj rulwkp #ru#wklv#wdvn1

Thank you!