
# My journey in IBM Data Science Course

Cesar Sussumu Ikarimoto

2022-July-16

# OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

# EXECUTIVE SUMMARY

- This presentation covers my development in Data Science, through the execution of modules from understanding the concepts and context related to data-driven mindset and using appropriate mathematical and computational tools, to presenting results which makes sense with some value to a specific audience.

# EXECUTIVE SUMMARY

- Topics
  - What is Data Science?
  - Tools for Data Science
  - Data Science Methodology
  - Python for Data Science, AI & Development
  - Python Project for Data Science
  - Databases and SQL for Data Science with Python
  - Data Analysis with Python
  - Data Visualization with Python
  - Machine Learning with Python
  - Applied Data Science Capstone

# INTRODUCTION

- In my journey in Data Science, I started with **module I of the program**, called *What is Data Science?*
- It was covered an introduction for the relevance of data, as a way of understanding and explaining some facts and additionally generating new insights. This is what defines a data-driven mindset.
- Besides the definition for data-driven analysis, notions for other terms like Big Data and Data Science were learned.
- The relevance of some soft skills that are expected for a data science student was showed, like curiosity, fluency in analytics and ability to communicate the findings.

# INTRODUCTION

- The **module II of the program** was Tools for Data Science
- The most important programming languages for Data Science were introduced, like:
  - Python
  - R
  - Scala
  - SQL
- Also, some open source tools:
  - GitHub
  - Jupyter Notebooks
  - Rstudio IDE
- And IBM tools for Data Science
  - Watson Studio

# INTRODUCTION

- The **module III:** Data Science Methodology
- Here the methodologies for Data Science were introduced, at a context of using data within the decision making process
- The stages of methodology for Data Science:
  - Business Understanding
  - Analytic Approach
  - Preparation
  - Modeling
  - Evaluating
  - Deployment
  - Feedback

# INTRODUCTION

- The **module IV:** Python for Data Science, AI & Development
- The module addressed Python language for Data Science with the following basic main topics:
  - Types
  - Expression and Variables
  - String Operations
  - List and Tuples
  - Dictionaries
  - Sets
  - Conditions and Branching
  - Loops
  - Functions
  - Exceptions handling
  - Objects and Classes

# INTRODUCTION

- Some Python libraries were introduced as well:
  - Pandas
  - Numpy
- And some tools for collecting data from other clients:
  - Simple APIs
  - REST APIs, Webscraping, and Working with Files

# INTRODUCTION

- **Module V:** Python Project for Data Science
- After completed the module IV, this module was intended to for demonstrating the skills learned previously
- The project is about creating a data science application, from gathering stock data;
- It was used Web Scraping, various Python libraries, and eventually, a dashboard

# INTRODUCTION

- **Module VI:** Databases and SQL for Data Science with Python
- This module consists of an introduction to SQL for Data Science
- Some basic concepts of this programming language and statements were introduced, like:
    - SELECT
    - COUNT, DISTINCT, LIMIT
    - INSERT
    - UPDATE
    - ETC…
- An introduction to creating a Database instance on Cloud and API was learned

# INTRODUCTION

- **Module VII:** Data Analysis with Python
- Here, how to analyze data using Python.
- How to prepare data, by selecting and filtering, cleaning, normalizing, to perform simple statistical analysis
- How to manipulate data frame
- Building machine learning regression models, and evaluate it using statistical tools
- How to visualize trends and graphics using Python
- Building data pipelines

# INTRODUCTION

- **Module VIII:** Data Visualization with Python
- Introduction to the importance of data visualization in Data Science
- Some tools were introduced:
  - Matplotlib
  - Seaborn
  - Folium
- And types of graphical objects:
  - Pie chart
  - Box plot
  - Scatter plot
  - Etc…

# INTRODUCTION

- **Module IX:** Machine Learning with Python
- In this module, we had an introduction to machine learning concepts and python libraries related to some ML algorithms
- Some supervised algorithms were addressed:
  - Linear and nonlinear Regression Models (simple and multiple)
  - Polynomial regression
  - K-Nearest Neighbours
  - Decision trees
  - Logistic regression
  - Support Vector Machine
- And unsupervised algorithms:
  - K-means Clustering
  - Hierarchical Clustering
- Finally, recommendation engines:
  - Content-based and collaborative filtering

# METHODOLOGY

- All projects were executed at Coursera website and:
  - IBM Watson Studio
  - Jupyter web
  - Skill Network Labs
  - Anaconda (local)

# METHODOLOGY

- For all activities data sets were available for collecting, treating, cleaning and standardizing
- Data sources available from:
- Module II
  - https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/auto.csv
  - https://archive.ics.uci.edu/ml/datasets/Automobile
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/automobileEDA.csv
- Module III
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0103EN-SkillsNetwork/labs/Module%202/recipes.csv
- Module IV
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Module%204/data/TopSellingAlbums.csv

# METHODOLOGY

- Data sources available from:
- Module V
  - https://aroussi.com/post/python-yahoo-finance
  - https://en.wikipedia.org/wiki/Florida
  - http://www.ibm.com
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/HTMLColorCodes.html
  - https://en.wikipedia.org/wiki/World_population
- Module VI
  - https://data.sfgov.org/Culture-and-Recreation/Film-Locations-in-San-Francisco/yitu-d5am?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDeveloperSkillsNetworkDB0201ENSkillsNetwork20127838-2022-01-01
- Module VII
  - https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data
- Module VIII
  - https://www.un.org/en/development/desa/population/migration/data/empirical2/migrationflows.asp
  - https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DV0101EN-SkillsNetwork/Data%20Files/Canada.xlsx

# METHODOLOGY

- In data collection and data wrangling/preparation, the following Python libraries were used:
  - Pandas
  - Yfinance
  - requests
  - BeautifulSoup
  - ibm_db

# METHODOLOGY

- Python libraries for EDA and data visualization:
  - Pandas
  - Numpy
  - Matplotlib
  - Seaborn
  - Folium

# METHODOLOGY

- Next, some libraries used for predictive analysis methodology
- Preprocessing
  - Sklearn
    - Preprocessing
    - Train_test_split
- Machine Learning basic Python libraries:
  - Sklearn
    - LinearRegression
    - KNeighborsClassifier
    - DecisionTreeClassifier
    - LogisticRegression
    - svm
  - Scipy
    - sigmoid

# RESULTS

# EDA – Exploratory Data Analysis

# EDA – Exploratory Data Analysis

In module III (Data Analysis with Python), it was required some analysis of data, specifically about House sales in king County, USA. Some kind of analysis consists in:

Data types of the attributes:

```
In [4]: df.dtypes

Out[4]: Unnamed: 0          int64
        id                  int64
        date               object
        price             float64
        bedrooms          float64
        bathrooms         float64
        sqft_living         int64
        sqft_lot            int64
        floors            float64
        waterfront          int64
        view                int64
        condition           int64
        grade               int64
        sqft_above          int64
        sqft_basement       int64
        yr_built            int64
        yr_renovated        int64
        zipcode             int64
        lat               float64
        long              float64
        sqft_living15       int64
        sqft_lot15          int64
        dtype: object
```

Data wrangling: dropping columns and checking some statistical information of the dataframe (using describe())

```
In [28]: df.drop(['id', 'Unnamed: 0'], axis=1, inplace=True)
         df.describe()
```

Out[28]:

| | price | bedrooms | bathrooms | sqft_living | sqft_lot | floors | waterfront |
|---|---|---|---|---|---|---|---|
| count | 2.161300e+04 | 21600.000000 | 21603.000000 | 21613.000000 | 2.161300e+04 | 21613.000000 | 21613.000000 |
| mean | 5.400881e+05 | 3.372870 | 2.115736 | 2079.899736 | 1.510697e+04 | 1.494309 | 0.007542 |
| std | 3.671272e+05 | 0.926657 | 0.768996 | 918.440897 | 4.142051e+04 | 0.539989 | 0.086517 |
| min | 7.500000e+04 | 1.000000 | 0.500000 | 290.000000 | 5.200000e+02 | 1.000000 | 0.000000 |
| 25% | 3.219500e+05 | 3.000000 | 1.750000 | 1427.000000 | 5.040000e+03 | 1.000000 | 0.000000 |
| 50% | 4.500000e+05 | 3.000000 | 2.250000 | 1910.000000 | 7.618000e+03 | 1.500000 | 0.000000 |
| 75% | 6.450000e+05 | 4.000000 | 2.500000 | 2550.000000 | 1.068800e+04 | 2.000000 | 0.000000 |
| max | 7.700000e+06 | 33.000000 | 8.000000 | 13540.000000 | 1.651359e+06 | 3.500000 | 1.000000 |

| | view | condition | grade | sqft_above | sqft_basement | yr_built | yr_renovated | zipc |
|---|---|---|---|---|---|---|---|---|
| count | 21613.000000 | 21613.000000 | 21613.000000 | 21613.000000 | 21613.000000 | 21613.000000 | 21613.000000 | 21613.00 |
| mean | 0.234303 | 3.409430 | 7.656873 | 1788.390691 | 291.509045 | 1971.005136 | 84.402258 | 98077.93 |
| std | 0.766318 | 0.650743 | 1.175459 | 828.090978 | 442.575043 | 29.373411 | 401.679240 | 53.50 |
| min | 0.000000 | 1.000000 | 1.000000 | 290.000000 | 0.000000 | 1900.000000 | 0.000000 | 98001.00 |
| 25% | 0.000000 | 3.000000 | 7.000000 | 1190.000000 | 0.000000 | 1951.000000 | 0.000000 | 98033.00 |
| 50% | 0.000000 | 3.000000 | 7.000000 | 1560.000000 | 0.000000 | 1975.000000 | 0.000000 | 98065.00 |
| 75% | 0.000000 | 4.000000 | 8.000000 | 2210.000000 | 560.000000 | 1997.000000 | 0.000000 | 98118.00 |
| max | 4.000000 | 5.000000 | 13.000000 | 9410.000000 | 4820.000000 | 2015.000000 | 2015.000000 | 98199.00 |

# EDA – Exploratory Data Analysis

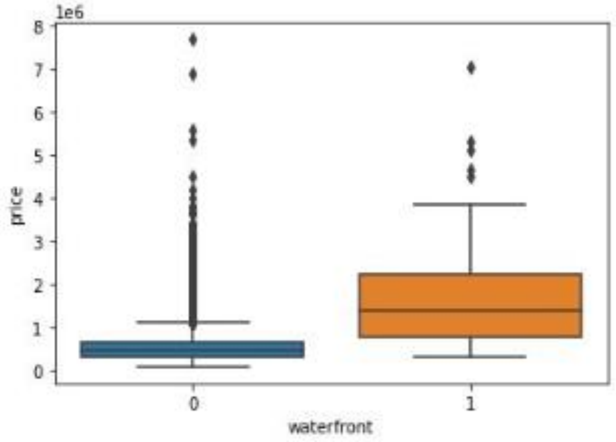
In module III (Data Analysis with Python), it was required some analysis of data, specifically about House sales in king County, USA. Some kind of analysis consists in:

Counting values from a specific attributes:

| | floors |
|---|---|
| 1.0 | 10680 |
| 2.0 | 8241 |
| 1.5 | 1910 |
| 3.0 | 613 |
| 2.5 | 161 |
| 3.5 | 8 |

Here, an example of graphical visualization to see some statistical information, like using a boxplot in seaborn

```
In [12]: sns.boxplot(x='waterfront', y='price', data=df)

Out[12]: <AxesSubplot:xlabel='waterfront', ylabel='price'>
```



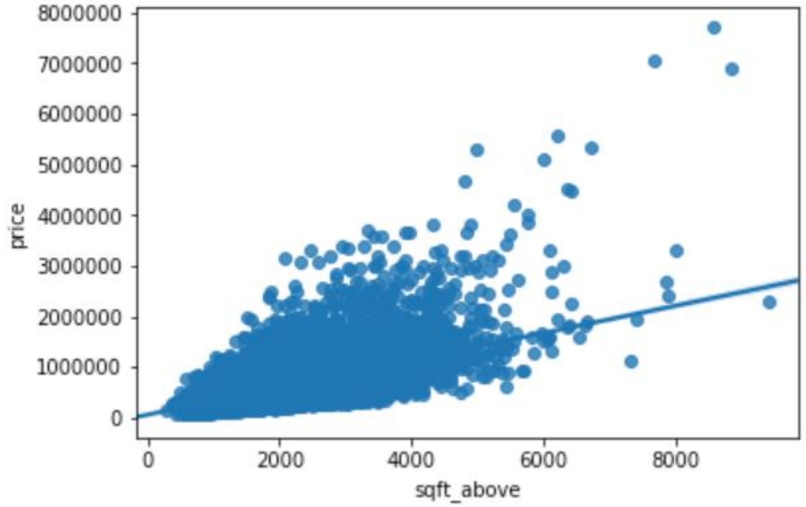

When we have 1 waterfront, the average price is greater than when we don't have a waterfront in the house. We can see, too, that for no-waterfront we have more outliers compared to with one waterfront

# EDA – Exploratory Data Analysis

In module III (Data Analysis with Python), it was required some analysis of data, specifically about House sales in king County, USA. Some kind of analysis consists in:

Or using a scatter plot, in a

Simple correlation of two attributes:




And checking if there is some good correlation of this two variables, using statistical tools like the LinearRegression.score

Fit a linear regression model to predict the 'price' using the feature 'sqft_living' then calculate the R^2. Take a screenshot of your code and the value of the R^2.

```
In [19]: X1 = df[['sqft_living']]
         Y1 = df['price']
         lm1 = LinearRegression()
         lm1.fit(X1,Y1)
         lm1.score(X1, Y1)

Out[19]: 0.4928532179037931
```

As we can see in the figure, apparently there is no good correlation between area in feet² and price; although we can trace a linear curve as showed, the statistical R² score of 0,49 points that it would not be a good model to use in this case

# EDA – Exploratory Data Analysis and SQL

IBM Developer

SKILLS NETWORK

# EDA – Exploratory Data Analysis and SQL

In module VI (Databases and SQL for Data Science with Python), one of the tasks was for analyse data from Socioeconomic indicators in Chicago, Chicago public schools and Chigago crime data. From getting the data to wrangling and analyse correlations for them:

Examples of exploring the data:

Finding the total number of crimes in Crime table:

```
[6]: %sql select count(*) from CHICAGO_CRIME_DATA

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[6]:

| 1 |
|---|
| 533 |

From socioeconomic indicators table, a query for areas with a per capita income lower than 11.000 dolar

```
[7]: %sql select COMMUNITY_AREA_NAME from CENSUS_DATA where PER_CAPITA_INCOME < 11000

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[7]:

| community_area_name |
|---|
| West Garfield Park |
| South Lawndale |
| Fuller Park |
| Riverdale |

From these SQL queries we can see the total number of crimes in Chicago area, as well the poorer areas

# EDA – Exploratory Data Analysis and SQL

In module VI (Databases and SQL for Data Science with Python), one of the tasks was for analyse data from Socioeconomic indicators in Chicago, Chicago public schools and Chigago crime data. From getting the data to wrangling and analyse correlations for them:

Examples of exploring the data:

Filtering the total number of crimes involving minors, in Chicago area

```
[10]: %sql select count(*) from CHICAGO_CRIME_DATA where DESCRIPTION like '%MINOR%'

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
[10]: 1
      2
```

And using the crime table yet, we can list all the cases of kidnapping crimes involving a child (using where , and , like statements)

```
[18]: %%sql
select * from CHICAGO_CRIME_DATA
where DESCRIPTION like '%CHILD%'
and PRIMARY_TYPE like '%KIDNA%'

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

| id | case_number | DATE | block | iucr | primary_type | description | location_description |
|---|---|---|---|---|---|---|---|
| 5276766 | HN144152 | 2007-01-26 | 050XX W VAN BUREN ST | 1792 | KIDNAPPING | CHILD ABDUCTION/STRANGER | STREET |

Using SQL statements as WHERE, AND, LIKE, we can refine the exploration of data

# EDA – Exploratory Data Analysis and SQL

In module VI (Databases and SQL for Data Science with Python), one of the tasks was for analyse data from Socioeconomic indicators in Chicago, Chicago public schools and Chigago crime data. From getting the data to wrangling and analyse correlations for them:

Examples of exploring the data:

Here an example of exploring the kind of crimes recorded at schools, using both Crime table and School table

```
[21]: %%sql
select LOCATION_DESCRIPTION, PRIMARY_TYPE, DESCRIPTION from CHICAGO_CRIME_DATA
where LOCATION_DESCRIPTION like '%SCHOOL%'

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[21]:

| location_description | primary_type | description |
|---|---|---|
| SCHOOL, PUBLIC, GROUNDS | BATTERY | SIMPLE |
| SCHOOL, PUBLIC, BUILDING | BATTERY | PRO EMP HANDS NO/MIN INJURY |
| SCHOOL, PUBLIC, BUILDING | BATTERY | SIMPLE |
| SCHOOL, PUBLIC, BUILDING | BATTERY | SIMPLE |
| SCHOOL, PUBLIC, GROUNDS | BATTERY | SIMPLE |
| SCHOOL, PUBLIC, GROUNDS | CRIMINAL DAMAGE | TO VEHICLE |

Grouping the most safe (elementary, middle or highschool) in Chicago area based the indicator SAFETY_SCORE

```
[71]: %%sql
select "Elementary, Middle, or High School"
, FLOAT(AVG(SAFETY_SCORE)) as Media_Score
from CHICAGO_PUBLIC_SCHOOLS
group by "Elementary, Middle, or High School"

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[71]:

| Elementary, Middle, or High School | media_score |
|---|---|
| ES | 49.0 |
| HS | 49.0 |
| MS | 48.0 |

Using SQL statements as GROUP BY, we can refine the exploration of data

# EDA – Exploratory Data Analysis and SQL

In module VI (Databases and SQL for Data Science with Python), one of the tasks was for analyse data from Socioeconomic indicators in Chicago, Chicago public schools and Chigago crime data. From getting the data to wrangling and analyse correlations for them:

Examples of exploring the data:

Here it is possible to analyse in order from the 5 highest to lowest % of households below poverty line in Chicago Area

```
[92]: %%sql
select  COMMUNITY_AREA_NAME, MAX(PERCENT_HOUSEHOLDS_BELOW_POVERTY) as Max_HH_Blw_Pov
from CENSUS_DATA
group by COMMUNITY_AREA_NAME
order by Max_HH_Blw_Pov DESC
fetch first 5 rows only

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[92]:

| community_area_name | max_hh_blw_pov |
|---|---|
| Riverdale | 56.5 |
| Fuller Park | 51.2 |
| Englewood | 46.6 |
| North Lawndale | 43.1 |
| East Garfield Park | 42.4 |

Which community area(number) is most crime prone

```
[96]: %%sql
select COMMUNITY_AREA_NUMBER, count(*) as Total_Crimes from CHICAGO_CRIME_DATA
group by COMMUNITY_AREA_NUMBER
order by Total_Crimes desc
nulls last limit 1

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[96]:

| community_area_number | total_crimes |
|---|---|
| 25 | 43 |

We can see that Riverdale is the lowest % area of households below poverty line in Chicago, and the community area number 25 has 43 total crimes computed.

# EDA – Exploratory Data Analysis and SQL

In module VI (Databases and SQL for Data Science with Python), one of the tasks was for analyse data from Socioeconomic indicators in Chicago, Chicago public schools and Chigago crime data. From getting the data to wrangling and analyse correlations for them:

Examples of exploring the data:

Using a sub-query to find the name of the community area with highest hardship index

```
[97]: %sql select COMMUNITY_AREA_NAME, HARDSHIP_INDEX from CENSUS_DATA\
      where HARDSHIP_INDEX = (select MAX(HARDSHIP_INDEX) from CENSUS_DATA)

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[97]:

| community_area_name | hardship_index |
|---|---|
| Riverdale | 98 |

And here, determining the Community Area Name with most number of crimes

```
[184]: %%sql
select COMMUNITY_AREA_NUMBER, COMMUNITY_AREA_NAME
from CENSUS_DATA
where COMMUNITY_AREA_NUMBER in (select count(COMMUNITY_AREA_NUMBER) as COUNT1 from
                                CHICAGO_CRIME_DATA
                                group by COMMUNITY_AREA_NUMBER
                                order by COUNT1 desc)

 * ibm_db_sa://pwl31778:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb;security=SSL
Done.
```

[184]:

| community_area_number | community_area_name |
|---|---|
| 1 | Rogers Park |
| 2 | West Ridge |
| 3 | Uptown |
| 4 | Lincoln Square |
| 5 | North Center |

We can see that Riverdale again, with the highest hardship index; however, Rogers Park is with the most crime rate of Chicago Area

# Data visualization – interactive maps with Folium

# Data visualization – interactive maps with Folium

In module X (Applied Data Science Capstone) we had exercises using folium library, analyzing data from SpaceX rocket launch and landing tables; some interesting information of launch sites, success rate and costs were addressed during the tasks:

```python
import folium
import wget
import pandas as pd
```

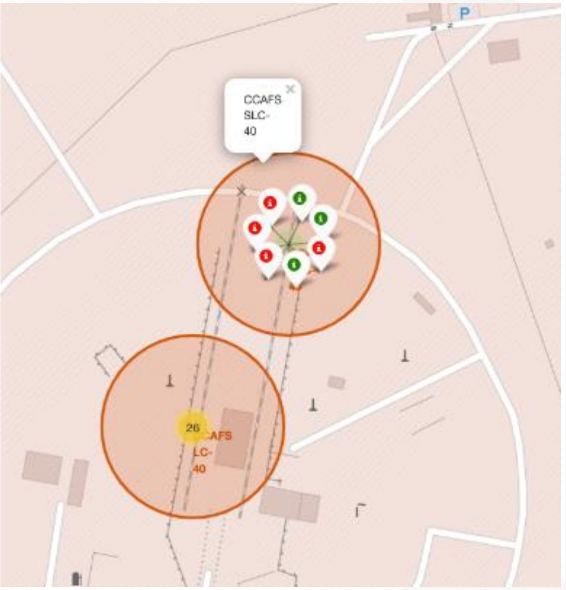
```python
# Import folium MarkerCluster plugin
from folium.plugins import MarkerCluster
# Import folium MousePosition plugin
from folium.plugins import MousePosition
# Import folium DivIcon plugin
from folium.features import DivIcon
```

```python
# Download and read the `spacex_launch_geo.csv`
spacex_csv_file = wget.download('https://cf-course
spacex_df=pd.read_csv(spacex_csv_file)
```

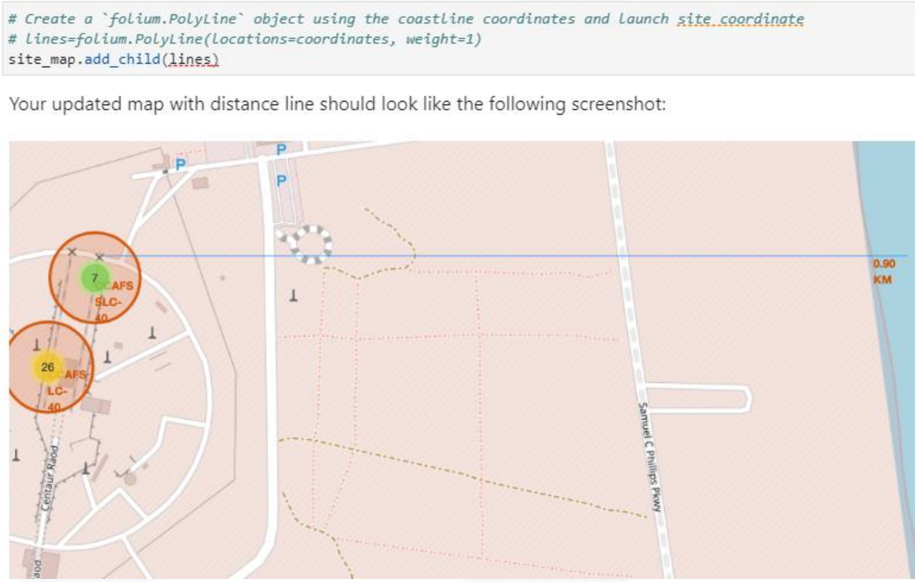
```python
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johns
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon 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)
```

# Data visualization – interactive maps with Folium

In module X (Applied Data Science Capstone) we had exercises using folium library, analyzing data from SpaceX rocket launch and landing tables; some interesting information of launch sites, success rate and costs were addressed during the tasks:







# Data visualization – Plotly Dash dashboard

# Data visualization – Plotly Dash dashboard

In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it’s used for making dashboards interactively with some features like user input and callback:

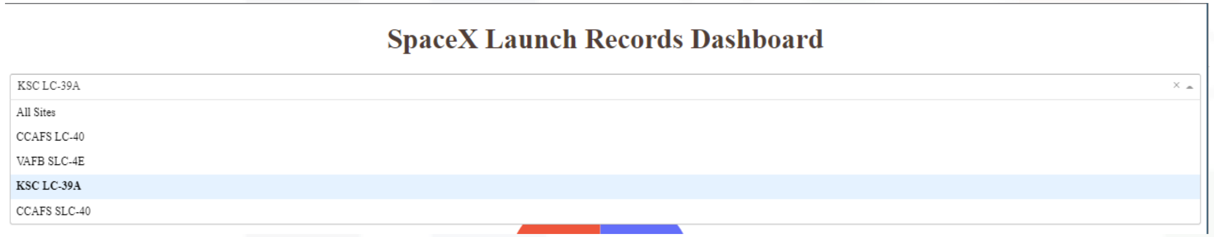
spacex_dash_app.py > ...

```python
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                dcc.Dropdown(id='site-dropdown',
                                             options=[
                                                 {'label': 'All Sites', 'value': 'ALL'},
                                                 {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
                                                 {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                                                 {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                                                 {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'}
```

Here is the code for creating a dropdown object with Launch Sites available for interactivity

# Data visualization – Plotly Dash dashboard

In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it’s used for making dashboards interactively with some features like user input and callback:




Here is the code for creating a dropdown object with Launch Sites available for interactivity

# Data visualization – Plotly Dash dashboard

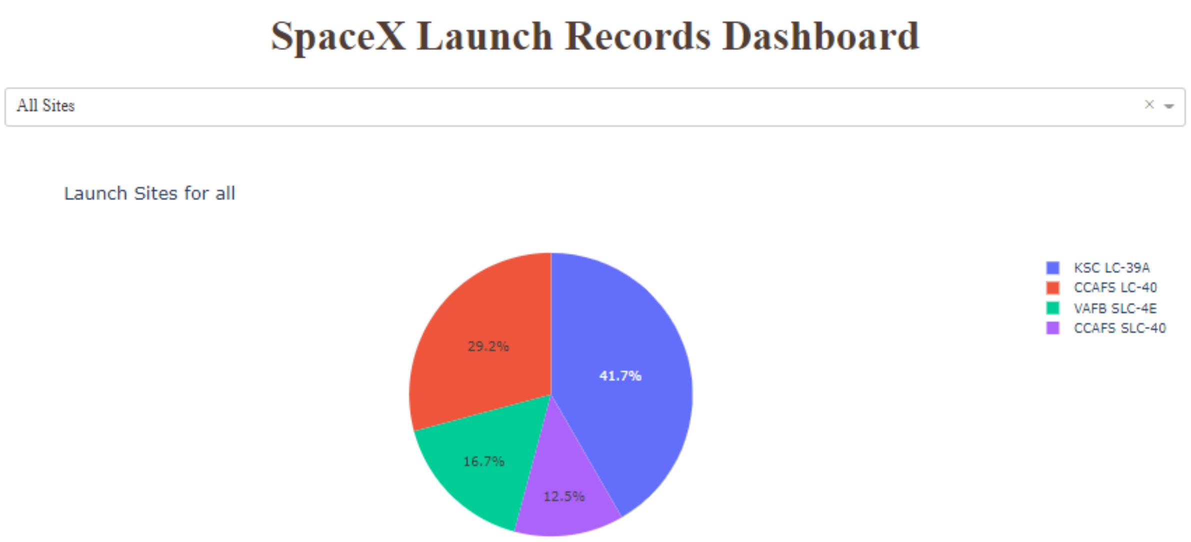
In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it's used for making dashboards interactively with some features like user input and callback:

```
# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(filtered_df, values='class',
        names='Launch Site',
        title='Launch Sites for all')
        return fig
    else:
        filtered_df=spacex_df[spacex_df['Launch Site']==entered_site]
        filtered_df=filtered_df.groupby(['Launch Site', 'class']).size().reset_index(name='class count')
        fig = px.pie(filtered_df, values='class count',
        names='class',
        title='Total Success Launches for Site')
        return fig
        # return the outcomes piechart for a selected site
```

Then, creating a callback function for a pie chart object

# Data visualization – Plotly Dash dashboard

In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it's used for making dashboards interactively with some features like user input and callback:




Then, creating a callback function for a pie chart object

# Data visualization – Plotly Dash dashboard

In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it’s used for making dashboards interactively with some features like user input and callback:

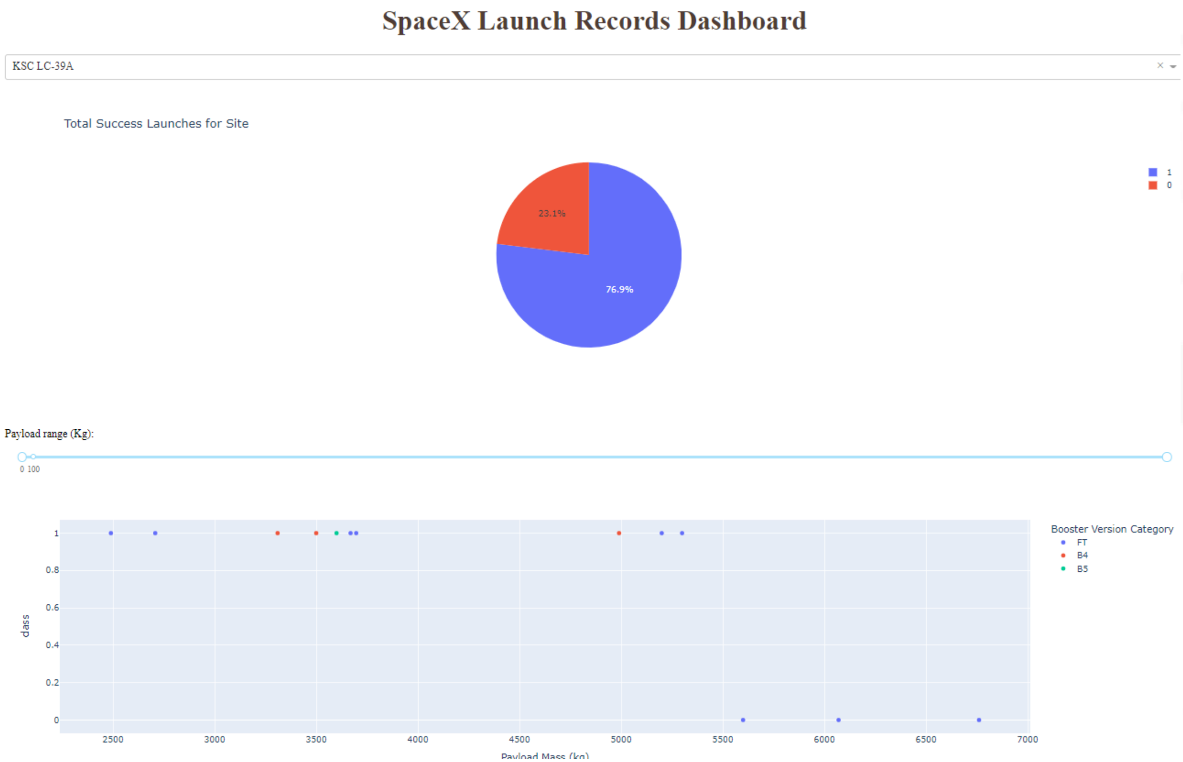
```
html.P("Payload range (Kg):"),
# TASK 3: Add a slider to select payload range
#dcc.RangeSlider(id='payload-slider',...)
dcc.RangeSlider(id='payload-slider',
                min=0, max=10000, step=1000,
                marks={0: '0',
                       100: '100'},
                value=[min_payload, max_payload]),

# TASK 4: Add a scatter chart to show the correlation between payload and launch
html.Div(dcc.Graph(id='success-payload-scatter-chart')),
])
```

```
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as

@app.callback(Output(component_id='success-payload-scatter-chart', component_property='figure'),
              [Input(component_id='site-dropdown', component_property='value'),Input(component_id='payload-slider
def get_scatter_chart(entered_site, payload_mass):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class',
        color='Booster Version Category'
        )
        return fig
    else:
        filtered_df=spacex_df[spacex_df['Launch Site']==entered_site]
        fig = px.scatter(filtered_df, x='Payload Mass (kg)', y='class',
        color='Booster Version Category'
        )
        return fig

# Run the app
if __name__ == '__main__':
    app.run_server()
```

Another object, a slider, inserted to the dash, as well as a scatter chart; both related to payload mass information

# Data visualization – Plotly Dash dashboard

In module X (Applied Data Science Capstone), it was introduced the Plotly visualization library, and Dash; it's used for making dashboards interactively with some features like user input and callback:




In scatter plot we have showed in color legend the booster version too

# Predictive analysis – Data classification

IBM Developer

SKILLS NETWORK

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

First of all, data collecting and wrangling, and EDA stage were done

**Load Data From CSV File**

In [52]:
```
df = pd.read_csv('loan_train.csv')
df.head()
```

Out[52]:

| | Unnamed: 0.1 | Unnamed: 0 | loan_status | Principal | terms | effective_date | due_date | age | education | Gender |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | PAIDOFF | 1000 | 30 | 9/8/2016 | 10/7/2016 | 45 | High School or Below | male |
| 1 | 2 | 2 | PAIDOFF | 1000 | 30 | 9/8/2016 | 10/7/2016 | 33 | Bechalor | female |
| 2 | 3 | 3 | PAIDOFF | 1000 | 15 | 9/8/2016 | 9/22/2016 | 27 | college | male |
| 3 | 4 | 4 | PAIDOFF | 1000 | 30 | 9/9/2016 | 10/8/2016 | 28 | college | female |
| 4 | 6 | 6 | PAIDOFF | 1000 | 30 | 9/9/2016 | 10/8/2016 | 29 | college | male |

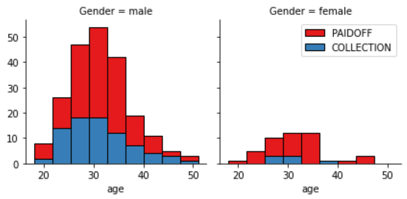
**Convert to date time object**

In [54]:
```
df['due_date'] = pd.to_datetime(df['due_date'])
df['effective_date'] = pd.to_datetime(df['effective_date'])
df.head()
```

Out[54]:

| | Unnamed: 0.1 | Unnamed: 0 | loan_status | Principal | terms | effective_date | due_date | age | education | Gender |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | PAIDOFF | 1000 | 30 | 2016-09-08 | 2016-10-07 | 45 | High School or Below | male |
| 1 | 2 | 2 | PAIDOFF | 1000 | 30 | 2016-09-08 | 2016-10-07 | 33 | Bechalor | female |
| 2 | 3 | 3 | PAIDOFF | 1000 | 15 | 2016-09-08 | 2016-09-22 | 27 | college | male |
| 3 | 4 | 4 | PAIDOFF | 1000 | 30 | 2016-09-09 | 2016-10-08 | 28 | college | female |
| 4 | 6 | 6 | PAIDOFF | 1000 | 30 | 2016-09-09 | 2016-10-08 | 29 | college | male |

In [57]:
```
bins = np.linspace(df.age.min(), df.age.max(), 10)
g = sns.FacetGrid(df, col="Gender", hue="loan_status", palette="Set1", col_wrap=2)
g.map(plt.hist, 'age', bins=bins, ec="k")

g.axes[-1].legend()
plt.show()
```




## One Hot Encoding

**How about education?**

In [62]:
```
df.groupby(['education'])['loan_status'].value_counts(normalize=True)
```

Out[62]:
```
education             loan_status
Bechalor              PAIDOFF        0.750000
                      COLLECTION     0.250000
High School or Below  PAIDOFF        0.741722
                      COLLECTION     0.258278
Master or Above       COLLECTION     0.500000
                      PAIDOFF        0.500000
college               PAIDOFF        0.765101
                      COLLECTION     0.234899
Name: loan_status, dtype: float64
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

Then, the normalize data action

## Normalize Data

Data Standardization give data zero mean and unit variance (technically should be done after train test split)

In [67]:

```
X= preprocessing.StandardScaler().fit_transform(X)
X[0:5]
```

Out[67]:

```
array([[ 0.51578458,  0.92071769,  2.33152555, -0.42056004, -1.20577805,
        -0.38170062,  1.13639374, -0.86968108],
       [ 0.51578458,  0.92071769,  0.34170148,  2.37778177, -1.20577805,
         2.61985426, -0.87997669, -0.86968108],
       [ 0.51578458, -0.95911111, -0.65321055, -0.42056004, -1.20577805,
        -0.38170062, -0.87997669,  1.14984679],
       [ 0.51578458,  0.92071769, -0.48739188,  2.37778177,  0.82934003,
        -0.38170062, -0.87997669,  1.14984679],
       [ 0.51578458,  0.92071769, -0.3215732 , -0.42056004,  0.82934003,
        -0.38170062, -0.87997669,  1.14984679]])
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

So, we started applying classification models

## K Nearest Neighbor (KNN)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state = np.random.seed())
print('Train set:', X_train.shape, y_train.shape)
print('Test set:', X_test.shape, y_test.shape)
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

best_accuracy=0.0
best_k=2

#looking for best k and accuracy

for k in range(2,10):
    neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train, y_train)
    yhat = neigh.predict(X_test)
    accuracy=metrics.accuracy_score(y_test, yhat)
    if accuracy>best_accuracy:
        best_accuracy=accuracy
        best_k=k
print('Best K is: ', best_k, 'Best accuracy is: ', best_accuracy)

#model for KNN Train

loan_status_KNN_train = KNeighborsClassifier(n_neighbors = 5).fit(X_train, y_train)
```

```
Best K is:  2 Best accuracy is:  0.7019230769230769
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

## Decision Tree

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
import sklearn.tree as tree

from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt

!conda install -c conda-forge pydotplus -y
!conda install -c conda-forge python-graphviz -y
```

```
#modeling the decision tree --> creating the instance loan_status_tree
loan_status_tree = DecisionTreeClassifier(criterion='entropy', max_depth=4)
#fitting the train set to this instance
loan_status_tree_train = loan_status_tree.fit(X_trainset, y_trainset)
```

```
# reading csv
df = pd.read_csv('loan_train.csv')
#converting due_date and effective_date to date type
df['due_date'] = pd.to_datetime(df['due_date'])
df['effective_date'] = pd.to_datetime(df['effective_date'])
#converting Gender to boolean
df['Gender'].replace(to_replace=['male','female'], value=[0,1],inplace=True)
#creating feature weekend
df['dayofweek'] = df['effective_date'].dt.dayofweek
df['weekend'] = df['dayofweek'].apply(lambda x: 1 if (x>3)  else 0)
#transforming each education to a boolean
Feature = df[['Principal','terms','age','Gender','weekend']]
Feature = pd.concat([Feature,pd.get_dummies(df['education'])], axis=1)
Feature.drop(['Master or Above'], axis = 1,inplace=True)
X = Feature
X= preprocessing.StandardScaler().fit_transform(X)

#creating y which is the target value
y = df['loan_status'].replace(to_replace=['PAIDOFF', 'COLLECTION'], value=[0,1]).values
X[0:5]
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

## Support Vector Machine

### Support Vector Machine

```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
%matplotlib inline
import matplotlib.pyplot as plt
```

```
#modeling SVM

from sklearn import svm
load_status_svm = svm.SVC(kernel='rbf', random_state=7)
load_status_svm_train = load_status_svm.fit(X_train, y_train)
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

## Logistic Regression

```
#changing target data to integer
#transforming strings of y to boolean
le_load_status = preprocessing.LabelEncoder()
le_load_status.fit(['PAIDOFF', 'COLLECTION'])
y = le_load_status.transform(y)
#normalize the data set
X=preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

```
#train and test the data set

print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)

#creating instance LogRegr
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LogRegr = LogisticRegression(C=0.01, solver='lbfgs').fit(X_train, y_train)
```

```
Train set: (242, 8) (242,)
Test set: (104, 8) (104,)
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

## Model evaluation using test set

For evaluation the best classification model, from sklearn some tools like jaccar_score, f1_score and log_loss were used

```
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
```

# Predictive analysis – Data classification

Module IX (Machine Learning with Python), addressed Machine learning with Python, using classification models like linear regression, K- nearest neighbor, logistic regression, decision tree, support vector machine :

## Model evaluation using test set

```
#predicting KNN model with test set
neigh = KNeighborsClassifier(n_neighbors = 3).fit(X_train, y_train)
yhat_KNN = neigh.predict(X_testset)

jaccard_KNN = jaccard_score(y_testset, yhat_KNN, pos_label=0)
f1score_KNN = f1_score(y_testset, yhat_KNN, pos_label=0)

#predicting decision tree model with test set
loan_status_tree_train = loan_status_tree.fit(X_trainset, y_trainset)
yhat_tree = loan_status_tree_train.predict(X_testset)

jaccard_tree = jaccard_score(y_testset, yhat_tree, pos_label=0)
f1score_tree = f1_score(y_testset, yhat_tree, pos_label=0)

#predicting svm model with test set
yhat_svm = load_status_svm.predict(X_testset)

jaccard_svm = jaccard_score(y_testset, yhat_svm, pos_label=0)
f1score_svm = f1_score(y_testset, yhat_svm, pos_label=0)

#predicting logistic regression with test set
yhat_LR = LogRegr.predict(X_testset)
yhat_LR_prob = LogRegr.predict_proba(X_testset)

jaccard_LR = jaccard_score(y_testset, yhat_LR, pos_label=0)
f1score_LR = f1_score(y_testset, yhat_LR, pos_label=0)
logloss_LR = log_loss(y_testset, yhat_LR_prob)

report = {'Algorithm': ['KNN', 'Decision Tree', 'SVM', 'LogisticRegression'],
          'Jaccard': [round(jaccard_KNN,3), round(jaccard_tree, 3), round(jaccard_svm, 3), round(jaccard_LR, 3)],
          'F1-score': [round(f1score_KNN, 3), round(f1score_tree, 3), round(f1score_svm, 3), round(f1score_LR, 3)],
          'LogLoss': ['NA', 'NA', 'NA', round(logloss_LR, 3)]}
report_df = pd.DataFrame.from_dict(report)
```

```
report_df.head()
```

| | Algorithm | Jaccard | F1-score | LogLoss |
|---|---|---|---|---|
| 0 | KNN | 0.500 | 0.667 | NA |
| 1 | Decision Tree | 0.667 | 0.800 | NA |
| 2 | SVM | 0.688 | 0.815 | NA |
| 3 | LogisticRegression | 0.647 | 0.786 | 0.599 |

# CONCLUSION

IBM Data Science Program, as an introduction to various subjects related to Mathematics, Statistics, Computer Science, Language Programming, among others, covered satisfactorily all first necessities.

The explanation of context was fullfilled, mainly in these days where the availability of a great amount of data, computer process velocity and critics mass are in an increasing rate, making possible using concepts of Machine Learning and Deep Learning models.

The focus on using Python as language, as well as SQL, and all the introduction to its basic use was very useful, since they are the most used programming language in Data Science. Introduction to JupyterLab and GitHub was very relevant, for the same reason.

All the methodology, from business needs in a macro perspective to deployment and use of models of Machine Learning, bring up a first contact to a data science project, opening up good perspectives for application in other challenges.