TRAINING CUM PROJECT REPORT 2020

BUILD A DATA SCIENCE WEB APP WITH STREAMLIT AND PYTHON

SUBMITTED BY

NAME: ANSHIK BANSAL
ENROLLMENT NUMBER: 01614807219

ABSTRACT

This project is about building a web app to observe the number of vehicle collisions happened in New York City, United States. The Motor Vehicle Collisions data tables contain information from all police reported motor vehicle collisions in NYC were got accessed from NYC Open Data public web portal.

Downloaded data in .csv format is then manipulated using pandas dataframe and simple Python script to load, explore, visualize and interact with data, and generate dashboards in less than 100 lines of Python code!

LEARNING CERTIFICATE

UNIVERSITY OF MICHIGAN

COURSE CERTIFICATE

05/27/2020

Anshik Bansal

has successfully completed

Programming for Everybody (Getting Started with Python)

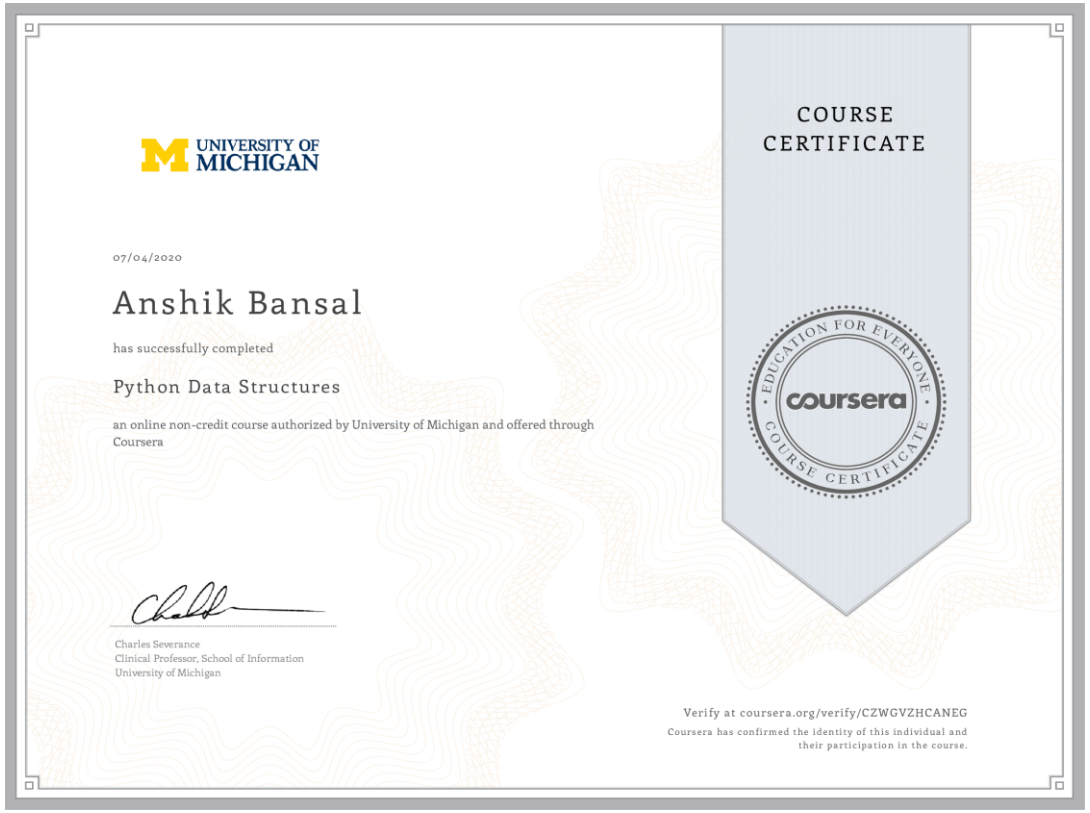
an online non-credit course authorized by University of Michigan and offered through Coursera

EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

Charles Severance
Clinical Professor, School of Information
University of Michigan

Verify at coursera.org/verify/USJ4UX9WXY9A

Coursera has confirmed the identity of this individual and their participation in the course.



UNIVERSITY OF MICHIGAN

COURSE CERTIFICATE

07/04/2020

Anshik Bansal

has successfully completed

Python Data Structures

an online non-credit course authorized by University of Michigan and offered through Coursera

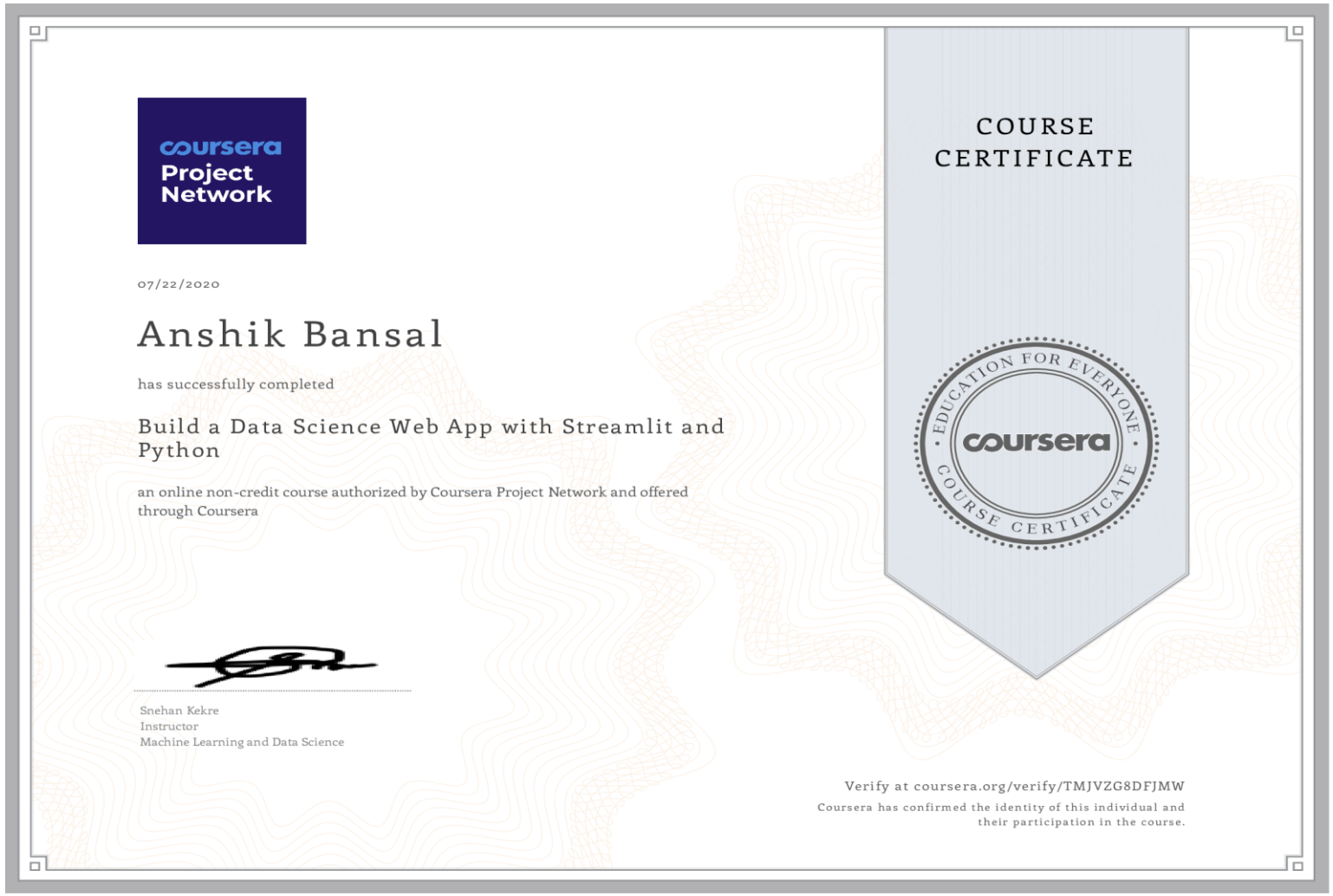
EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

Charles Severance
Clinical Professor, School of Information
University of Michigan

Verify at coursera.org/verify/CZWGVZHCANEG

Coursera has confirmed the identity of this individual and their participation in the course.

GUIDED PROJECT CERTIFICATE



coursera Project Network

07/22/2020

Anshik Bansal

has successfully completed

Build a Data Science Web App with Streamlit and Python

an online non-credit course authorized by Coursera Project Network and offered through Coursera

Snehan Kekre
Instructor
Machine Learning and Data Science

COURSE CERTIFICATE

EDUCATION FOR EVERYONE · COURSE CERTIFICATE ·

coursera

Verify at coursera.org/verify/TMJVZG8DFJMW

Coursera has confirmed the identity of this individual and their participation in the course.

TRAINING REPORT

PROGRAMMING FOR EVERYBODY (GETTING STARTED WITH PYTHON)

DURATION: APRIL 25, 2020 TO MAY 27, 2020

KEY LEARNINGS

- Installing Python and writing down first program in Python
- Learning the basics of the Python programming language
- Using variables to store, retrieve and calculate information
- Utilize core programming tools such as functions and loops

PYTHON DATA STRUCTURES

DURATION: JUNE 1, 2020 TO JULY 4, 2020

KEY LEARNINGS

- Learning the principles of common data structures & how they are used
- Creating programs that are able to read and write data from files
- Storing data as key/value pairs using Python dictionaries
- Accomplish multi-step tasks like sorting or looping using tuples

TRAINING CUM PROJECT REPORT 2020

PROJECT REPORT

BUILD A DATA SCIENCE WEB APP WITH STREAMLIT AND PYTHON

DURATION: JULY 22, 2020

KEY LEARNINGS

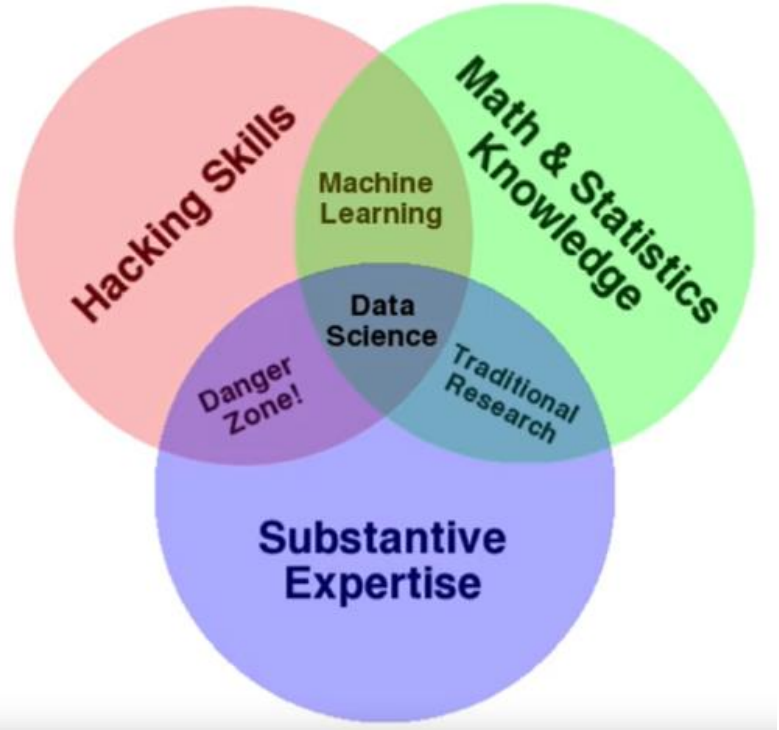
- Build interactive web applications with Streamlit and Python
- Using Pandas for data manipulation in data science workflows
- Using pydeck to create interactive deck.gl maps

INTRODUCTION TO DATA SCIENCE

The history of data science goes back a little further than 2004, which is where the Google search term history begins. But this, at least, gives a sense for how popular the area is now.

The techniques and methodologies of data science stem from the fields of computer science and statistics. One of the most well cited diagrams describing the field comes from Drew Conway where he suggested data science is the intersection of hacking skills, math and stats knowledge, and substantial expertise. This diagram might be a bit of an oversimplification, but I think it's a great start.

Data science is definitely one of those areas where you ask ten people and get ten different answers.



INTRODUCTION TO DATA SCIENCE

- The hacking skills refer to the computer science skills. Data is digital. In order to efficiently manipulate the data, you need to have some programming skills. You need to be comfortable at the command line, be able to manipulate files of different formats, program algorithms that will modify the data, etc.
- The Math and Statistics knowledge refer to the mathematical abilities required to properly analyze and infer things about the data.
- The Substantive Knowledge is the knowledge specific to the area where data science is applied. It is often referred to as "domain knowledge". For example, if you are applying data science to genome problems, you should have "substantive knowledge" on that topic.

PYTHON LIBRARIES USED IN PROJECT

NUMPY

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

You can get started using pydeck with a simple copy and paste in your terminal:

pip install numpy

PYTHON LIBRARIES USED IN PROJECT

PANDAS

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tool build on the top of Python programming language.

When working with tabular data, such as data stored in spreadsheets or databases, pandas is the right tool for you. pandas will help you to explore, clean and process your data. In pandas, a data table is called a DataFrame.

pandas support the integration with many file formats or data sources out of the box (csv, excel, sql, json, parquet, etc.).

You can get started using pydeck with a simple copy and paste in your terminal:

pip install pandas

PYTHON LIBRARIES USED IN PROJECT

PYDECK

The pydeck library is a set of Python bindings for making large-scale geospatial visualizations with deck.gl, optimized for a Jupyter environment.

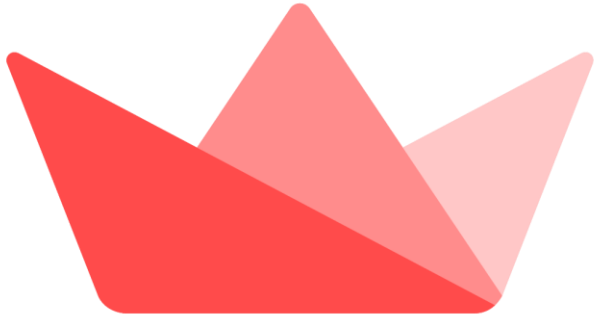
The mission of pydeck is to let Python users create deck.gl maps without having to know a lot of Javascript. At its core, pydeck focuses on data analytics use cases, so it works best in environments used by data professionals, like Jupyter Notebooks. With pydeck, users can embed visualizations interactively in a Jupyter Notebook or simply export them to a standalone HTML file.

You can get started using pydeck with a simple copy and paste in your terminal:

pip install pydeck

PYTHON LIBRARIES USED IN PROJECT

STREAMLIT



Streamlit is a tool to create Web-based frontends with a focus for the Machine Learning scientist or engineer. The Streamlit enabled Machine Learning scientist to deploy scripts and web data applications without using Flask, Django, or other tools.

Streamlit watches for changes on each save and updates the app live while you're coding. Code runs from top to bottom, always from a clean state, and with no need for callbacks. It's a simple and powerful app model that lets you build rich UIs incredibly quickly.

You can get started using streamlit with a simple copy and paste in your terminal:

pip install streamlit

ADVANTAGES OF STREAMLIT

A few of the advantages of using Streamlit tools like Dash and Flask includes:

1. It embraces Python scripting; No HTML, CSS & Javascript knowledge is needed!
2. Less code is needed to create a beautiful application
3. No callbacks are needed since widgets are treated as variables
4. Data caching simplifies and speeds up computation pipelines.

PROJECT IMPLEMENTATION

Part 1: Writing Title Text

To write text in Streamlit, just use function such as st.title, st.markdown, st.subtitle and you're ready to built a web app with no more required knowledge in other languages.

Every good app has a title, so let's just add one:

```
import streamlit as st

st.title("Motor Vehicle Collisions in New York City")
st.markdown("This web application is a Streamlit dashboard that can be used "
            "to analyze Motor Vehicle Collisions in NYC, US 👆💥🚗")
```

PROJECT IMPLEMENTATION

Part 2: Reading raw data CSV format file

To read the data, we used pandas library:

```
import pandas as pd
DATA_URL = (
    "./Motor_Vehicle_Collisions_-_Crashes.csv"
)
data = pd.read_csv(DATA_URL, nrows=nrows, parse_dates=[['CRASH_DATE', 'CRASH_TIME']])
```

PROJECT IMPLEMENTATION

Part 3: Fetching Some Data

load_data is a plain old function that downloads some data, puts it in a Pandas dataframe, drop the rows that doesn't have either latitude or longitude and converts the date column from text to datetime. The function accepts a single parameter (nrows), which specifies the number of rows that you want to load into the DataFrame.

```
def load_data(nrows):
    data = pd.read_csv(DATA_URL, nrows=nrows, parse_dates=[['CRASH_DATE', 'CRASH_TIME']])
    data.dropna(subset=['LATITUDE', 'LONGITUDE'], inplace=True)
    lowercase = lambda x: str(x).lower()
    data.rename(lowercase, axis="columns", inplace=True)
    data.rename(columns={"crash_date_crash_time": "date/time"}, inplace=True)
    #data = data[['date/time', 'latitude', 'longitude']]
    return data
```

PROJECT IMPLEMENTATION

Part 4: Loading The Data

To load the data, we'll just invoke the function with parameter defining the number of rows we needed to load into the pandas DataFrame.

```
# Load 10,000 rows of data into the dataframe.
data = load_data(10000)
```

Part 5: Caching Data

Streamlit provides a caching mechanism that allows our app to stay performant even when loading data from the web, manipulating large datasets, or performing expensive computations. This is done with the @st.cache decorator.

```
@st.cache(persist=True)
```

PROJECT IMPLEMENTATION

Part 6: Creating Slider & Map

To create a slider we just need to pass the slider range with function st.slider. The first value is the minimum range, whilst the other value represents the maximum range.

To create a map, we need to invoke st.map command. In our case, we have set data query equal to the slider value and let map function plot the data on geographical coordinates using latitude and longitude value parsed from the csv data.

```
st.header("Where are the most people injured in NYC?")
injured_people = st.slider("Number of persons injured in vehicle collisions", 0, 19)
st.map(data.query("injured_persons >= @injured_people")[["latitude", "longitude"]].dropna(how="any"))
```

PROJECT IMPLEMENTATION

Part 7: Using Numpy & pydeck

In this, we use numpy to locate the initial map location, so as to avoid mouse-scrolling to view the data on the map. For this, we first find the average of the latitude and longitude values respectively using numpy and then assign those values stored in tuple using pydeck. We can also change zoom level according to our need.

To produce the data, here we use pydeck layer which sits on the top of the Mapbox Map. We can further adjust options such as extruded, radius etc. to represent and visualize data better.

```
st.header("Where are the most people injured in NYC?")
injured_people = st.slider("Number of persons injured in vehicle collisions", 0, 19)
st.map(data.query("injured_persons >= @injured_people")[["latitude", "longitude"]].dropna(how="any"))
```

PROJECT IMPLEMENTATION

```
midpoint = (np.average(data["latitude"]), np.average(data["longitude"]))
st.write(pdk.Deck(
    map_style="mapbox://styles/mapbox/light-v9",
    initial_view_state={
        "latitude": midpoint[0],
        "longitude": midpoint[1],
        "zoom": 11,
        "pitch": 50,
    },
    layers=[
        pdk.Layer(
        "HexagonLayer",
        data=data[['date/time', 'latitude', 'longitude']],
        get_position=["longitude", "latitude"],
        auto_highlight=True,
        radius=100,
        extruded=True,
        pickable=True,
        elevation_scale=4,
        elevation_range=[0, 1000],
        ),
    ],
))
```

OUTPUTS

Where are the most people injured in NYC?

Number of persons injured in vehicle collisions

3

0 19

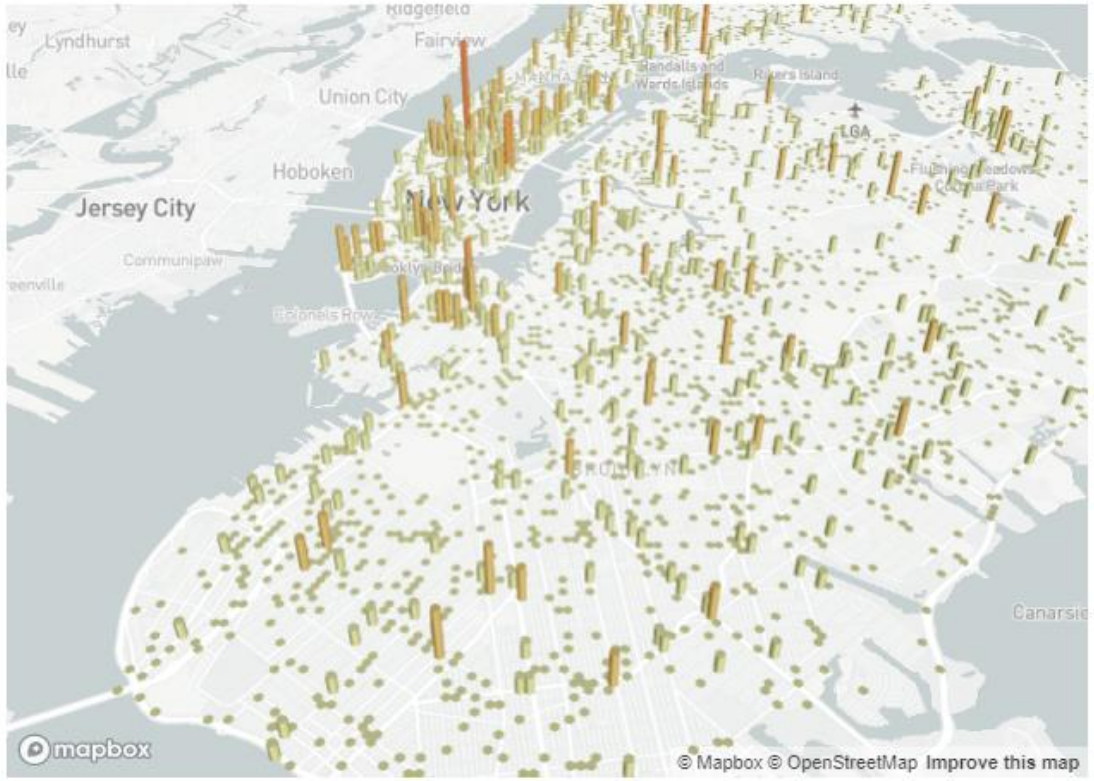
How many collisions occur during a given time of day?

Hour to look at

11

0 23

Vehicle collisions between 11:00 and 12:00



OUTPUTS

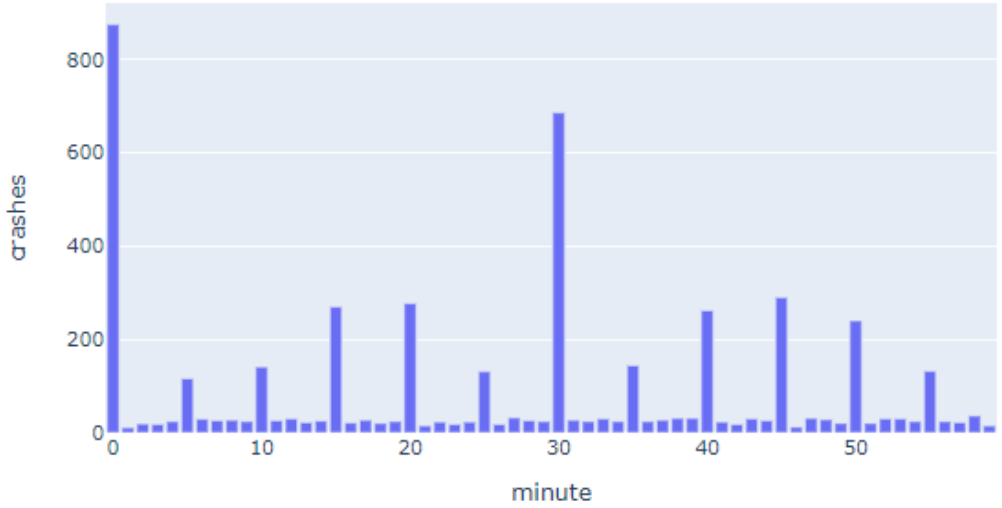
TRAINING CUM PROJECT REPORT 2020

Show raw data

Raw data by minute between 11:00 and 12:00

	date/time	borough	zip_code	latitude	longitude	
51	Jul 30, 2018 11:00 AM	nan	NaN	40.7923	-73.7933	POIN
141	Mar 18, 2020 11:17 AM	BROOKLYN	11220	40.6393	-74.0037	POIN
236	Aug 15, 2016 11:50 AM	MANHATTAN	10012	40.7220	-73.9963	POIN
249	Aug 17, 2016 11:30 AM	BRONX	10452	40.8320	-73.9235	POIN
263	Aug 13, 2016 11:30 AM	QUEENS	11373	40.7437	-73.8838	POIN
268	Jul 27, 2016 11:40 AM	BROOKLYN	11212	40.6546	-73.9220	POIN
290	Aug 3, 2016 11:30 AM	BRONX	10454	40.8043	-73.9216	POIN
307	Aug 6, 2016 11:00 AM	QUEENS	11423	40.7197	-73.7643	POIN
336	Aug 12, 2016 11:00 AM	MANHATTAN	10003	40.7244	-73.9907	POIN
362	Aug 16, 2016 11:00 AM	nan	NaN	40.7427	-73.9541	POIN
364	Aug 7, 2016 11:30 AM	nan	NaN	40.7543	-73.8975	POIN

Breakdown by minute between 11:00 and 12:00



OUTPUTS

Top 5 dangerous streets by affected class

Affected class

Motorists

	on_street_name	injured_motorists
4517	BRUCKNER EXPRESSWAY	18
98151	BREWER BOULEVARD	13
17405	ROCKAWAY AVENUE	13
35456	EAST 126 STREET	11
53118	BRUCKNER EXPRESSWAY	10

RESULT

Streamlit has democratized the whole process to create apps, and it's one of the most recommended Python Library for data science web apps. In this project, we created a simple data science web app. But the possibilities are endless.

We can provide real-time data in format supported by Pandas Dataframe, that can improve the end results displayed on the web application.

The web app data can also be used to improve the road congestion problems, and can be very helpful to reduce the number of vehicle collisions and accidents.

Such visualized data can be made publicly available to a web address, so that the individuals can get to know the risk factors on choosing their vehicle pathway, and it can drastically reduce the jamming and accident issues.

THANK YOU ☺