Data Scientist Take Home Challenge

311 is a service that New York City residents can use to make non-emergency reports to the city for things like noise complaints, graffiti, potholes, etc. Each complaint that comes into the city is registered, along with additional information like the location, the neighborhood, the agency that responded to the request, etc.

About the dataset

The data has 40+ columns and more than 21,000,000 rows, and a total size of roughly 12 GB. To minimize the impact on your computer, we've stored the data on a Redshift database and included a python function to connect to the database below. If you'd prefer to work with the raw data in flat files, you can download it yourself here: https://nycopendata.socrata.com/Social-Services/311-Service-Requests-from-2010-to-Present/erm2-nwe9.

This data set is large. We work with even larger data sets all the time here at Ezoic, and employ a few different methods of dealing with them. Feel free to use things like sampling, streaming, etc to be able to manage the data set while still coming up with valid conclusions.

The data has the following columns:

- unique_key: Unique identifier of a Service Request (SR) in the open data set
- created_date: Date SR was created
- closed_date: Date SR was closed
- agency: Acronym of responding City Government Agency
- agency_name: Full Agency name of responding City Government Agency
- complaint_type: This is the first level of a hierarchy identifying the topic of the incident or condition. Complaint Type may have a corresponding Descriptor (below) or may stand alone.
- descriptor: This is associated to the Complaint Type, and provides further detail on the incident or condition. Descriptor values are dependent on the Complaint Type, and are not always required in SR.

- location_type: Describes the type of location used in the address information
- incident_zip: Incident location zip code, provided by geo validation.
- incident_address: House number of incident address provided by submitter.
- street_name: Street name of incident address provided by the submitter
- cross_street_1: First Cross street based on the geo validated incident location
- cross_street_2: Second Cross Street based on the geo validated incident location
- intersection_street_1: First intersecting street based on geo validated incident location
- intersection_street_2: Second intersecting street based on geo validated incident location
- address_type: Type of incident location information available.
- city: City of the incident location provided by geovalidation.
- landmark: If the incident location is identified as a Landmark the name of the landmark will display here
- facility_type: If available, this field describes the type of city facility associated to the SR
- status: Status of SR submitted
- due_date: Date when responding agency is expected to update the SR. This is based on the Complaint Type and internal Service Level Agreements (SLAs).
- resolution_description: Describes the last action taken on the SR by the responding agency. May describe next or future steps.
- resolution_action_updated_date: Date when responding agency last updated the SR.
- community_board: Provided by geovalidation.
- bbl: Borough Block and Lot, provided by geovalidation. Parcel number to identify the location of location of buildings and properties in NYC.
- borough: Provided by the submitter and confirmed by geovalidation.
- x_coordinate_state_plane: Geo validated, X coordinate of the incident location.
- y_coordinate_state_plane: Geo validated, Y coordinate of the incident location.
- open_data_channel_type: Indicates how the SR was submitted to 311. i.e. By Phone, Online, Mobile, Other or Unknown.
- park_facility_name: If the incident location is a Parks Dept facility, the Name of the facility will appear here
- park_borough: The borough of incident if it is a Parks Dept facility
- vehicle_type: If the incident is a taxi, this field describes the type of TLC vehicle.
- taxi_company_borough: If the incident is identified as a taxi, this field will display the borough of the taxi company.
- taxi_pick_up_location: If the incident is identified as a taxi, this field displays the taxi pick up location
- bridge_highway_name: If the incident is identified as a Bridge/Highway, the name will be displayed here.
- bridge_highway_direction: If the incident is identified as a Bridge/Highway, the direction where the issue took place would be displayed here.
- road_ramp: If the incident location was Bridge/Highway this column differentiates if the issue was on the Road or the Ramp.

- bridge_highway_segment: Additional information on the section of the Bridge/Highway were the incident took place.
- latitude: Geo based Lat of the incident location
- longitude: Geo based Long of the incident location
- location: Combination of the geo based lat & long of the incident location

Connecting to the Database

This function will connect you to our 311 database. You might need to install pandas and psycopg2 to run.

```
import pandas as pd

import psycopg2


def query_db(query):

    '''

    Runs sql query on 311 database and returns a pandas DataFrame.

    Redshift is a data warehouse based on PostgreSQL, so syntax is mostly the same

    '''

    host = 'interview-ds.ckgnwnm6pw4o.us-east-1.redshift.amazonaws.com'

    port = 5439

    db = 'interview'

    username = 'dsguest'

    password = 'nX9EFYUZ5Yu#0q'

    conn = psycopg2.connect(host=host, port=port, dbname=db, user=username, password=password)
```

```
    cur = conn.cursor()
    cur.execute(query)
    rows = cur.fetchall()
    rows = pd.DataFrame(rows)
    return rows


# Example usage:
print(query_db("SELECT count(1) FROM public.three_one_one;"))
```

Packages you might need need:

Psycopg2 may require some extra libraries to be installed. Information on how to install psycopg2 can be found here: http://initd.org/psycopg/docs/install.html

If not already installed, run this from your command line:

```
pip install pandas
pip install psycopg2
```

Challenge One: Data Analysis

1. How many incidents were opened in each year (beginning with 2010)?
2. Which borough has the most incidents?

Challenge Two: Build a Data Product

Your task is to use your data science skills to identify a data story and build a data product from the New York City 311 Service Request data set. First, we'd like you to create a model to predict what the type of a given complaint will be. After that, we encourage you to add anything else you find exciting that showcases your skills: another predictive model, an interactive dashboard or visualization, an interesting statistical analysis, or anything else you dream up! Be creative, but the most important thing is to showcase your data skills.

How you will be evaluated:

Methodology: Is the analysis sound? If you make a model, how is it validated?

Code: Is it clean, understandable, and efficient?

Communication: Does your project make sense to both technical and nontechnical audience?

We are most interested in seeing how you think, how you work through problems, and the conclusions you can draw and value you can derive from data. You won't be penalized if your hypotheses don't reveal anything interesting or revolutionary in the data. Focus on the approach rather than optimizing for maximal accuracy, we don’t care how much processing power you’re able to use. We want to see how you came to your conclusions and what you've built from the data.

When you’re done, email your finished product back to us, complete with all of the code you used. We’ve allocated 7 days for this project, but we don’t expect you to put in 7 days worth of work. This limit is there to help move the interview process along and give you a date to work towards, rather than an indication of our expectations.

Don’t hesitate to reach out if you have any questions!