

Homework 4: Big Data and Google Cloud

UIC CS 418, Spring 2022

According to the **Academic Integrity Policy** of this course, all work submitted for grading must be done individually, unless otherwise specified. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you cannot work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at <https://dos.uic.edu/conductforstudents.shtml> (<https://dos.uic.edu/conductforstudents.shtml>).

This homework is an individual assignment for all graduate students. Undergraduate students are allowed to work in pairs and submit one homework assignment per pair. There will be no extra credit given to undergraduate students who choose to work alone. The pairs of students who choose to work together and submit one homework assignment together still need to abide by the Academic Integrity Policy and not share or receive help from others (except each other).

Due Date

This assignment is due at 11:59pm on April 15th, 2022.

Instructions

You need to complete all code and answer all questions denoted by **Q#** in this notebook. When you are done, you should export **hw4.ipynb** with your answers as a PDF file, upload that file **hw4.pdf** to *Homework 4 - written* on Gradescope, tagging each question.

Select your submission subdirectory **sub** and your completed Jupyter notebook (**hw4.ipynb** file) and zip it. As a result your **sub** subdirectory and **hw4.ipynb** file should be in the root of your zip file. Upload this zip file to *Homework 4 - code* on Gradescope.

For undergraduate students who work in a team of two, only one student needs to submit the homework and just tag the other student on Gradescope.

Keep an eye out for the following icons:

 Specifies that you need to add something to the notebook in order to get credit.

 Specifies that you need to save something to **sub** (submission) subdirectory or verify the existence of some file in **sub** to get credit.

 Warnings to avoid common mistakes and pitfalls. Pay special attention to these.

Autograding

Q3 and Q4 will be graded based on your PDF submissions, the rest of the questions will be graded using an Autograder. All parts of Homework 4 are graded based on correctness, **not** based on completion.

Before We Start

Before we dive into the assignment, we need to install the following python packages:

- seaborn*
- matplotlib*
- nltk*
- sklearn*
- pandas*
- numpy*
- google-cloud
- google-cloud-storage
- google-cloud-bigquery[pandas]

Packages marked with * should be pre-installed in a conda environment. You may install missing packages one at a time using `pip` command or you can run the following shell command to install them all at once:

```
In [1]: !pip install -r ./requirements.txt > pip-log.txt
```

 If you are installing using the shell command listed above then pay attention to any warnings and check the logs `pip-log.txt`.

Now, we are ready to import all dependencies.

```
In [2]: from google.cloud import storage
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
import pandas as pd
import numpy as np
import os

%load_ext google.cloud.bigquery
```

We also need to ensure that we have required nltk packages installed:

```
In [3]: nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

[nltk_data] Downloading package stopwords to
[nltk_data]      /Users/malika/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/malika/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]      /Users/malika/nltk_data...
[nltk_data] Package averaged_perceptron_tagger is already up-to-
[nltk_data]      date!
```

```
Out[3]: True
```

Now that we have everything we need, we can dive into the homework itself.

Cloud Services

Most real world systems in the modern world generate copious amount of data (billions of records) and in order to process this data and to enable reasoning through statistical models, we need high computational power and memory which is not available in most personal machiens.

One solution is to build and maintain your own clusters of high-end machines which has a massive overhead and cost associated with it, which is definitely not suitable for individuals or small teams. Hence, most individuals and companies rely on cloud service providers for a range of services, including but not limited to storage, processing, querying and visualizing large datasets.

In this homework, we will use [Google Cloud \(<https://cloud.google.com/>\)](https://cloud.google.com/) to query and visualize data and we will train and deploy a machine learning model in the cloud. The rest of the homework is divided in 4 parts.

Part 0: Setup: Initial configurations

Part 1: Big Query: A database for Big Data

Part 2: Data Studio: A visualization tool for Big Data

Part 3: AI Platform: Machine Learning with Big Data

These services can save a lot of time and make your life easier when handling a large amount of data which cannot be processed using a single machine. So let's get started!

Part 0: Setup

Setup Billing

Follow this [link \(<https://gcp.secure.force.com/GCPEDU?cid=%2Fos74HWCYOoMZ6Cna61Hfgre13ONNcYNIGbOYN9GjPcp5KzV%2BVf92ePVq9G9pYY6/>\)](https://gcp.secure.force.com/GCPEDU?cid=%2Fos74HWCYOoMZ6Cna61Hfgre13ONNcYNIGbOYN9GjPcp5KzV%2BVf92ePVq9G9pYY6/) to claim \$50 credit using your @uic.edu email. This homework can be completed while utilizing less than \$5. Fill up the details in following form and it will send you an email to verify your account.

Cloud Platform Education Grants

Use credits provided to you via the Google Cloud Platform Education Grants program to access Google Cloud Platform. Get what you need to build and run your apps, websites and services.

Thank you for your interest in Google Cloud Platform Education Grants. Please fill out the form below to receive a coupon code for credit to use on Google Cloud Platform.

First Name

Last Name

School Email

 @uic.edu

If you do not see your domain listed, please contact your course instructor: ezheleva@uic.edu

By clicking "Submit" below, you agree that we may share the following information with your educational institution and course instructor (ezheleva@uic.edu): (1) personal information that you provide to us on this form and (2) information regarding your use of the coupon and Google Cloud Platform products.

Submit

[Privacy Policy](#)

After you have received your verification email (shown below), follow the link in email to request a coupon



Dear [REDACTED]

Thank you for your interest in downloading a Google Cloud Platform Coupon Code. Please click on this [link](#) to verify your email address and a code will be sent to your email account.

Instructor Name: [Elena Zheleva](#)

Email Address: ezheleva@uic.edu

School: [University of Illinois at Chicago](#)

Course/project: CS 418 Introduction to Data Science

If you have any questions, please contact your course instructor as listed above.

Thanks,

Google Cloud Platform Education Grants Team

You will receive a coupon in email as shown below, follow the link to redeem it.

Be careful to redeem the coupon to your Google account with UIC email NOT your personal account.



Dear [REDACTED]

Here is your Google Cloud Coupon Code: [REDACTED] E959-6E3K

Click [\[here\]](#) to redeem.

Course/Project Information

Instructor Name: [Elena Zheleva](#)

Email Address: ezheleva@uic.edu

School: [University of Illinois at Chicago](#)

Course/project: [CS 418 Introduction to Data Science](#)

Activation Date: [1/11/2022](#)

Redeem By: [5/11/2022](#)

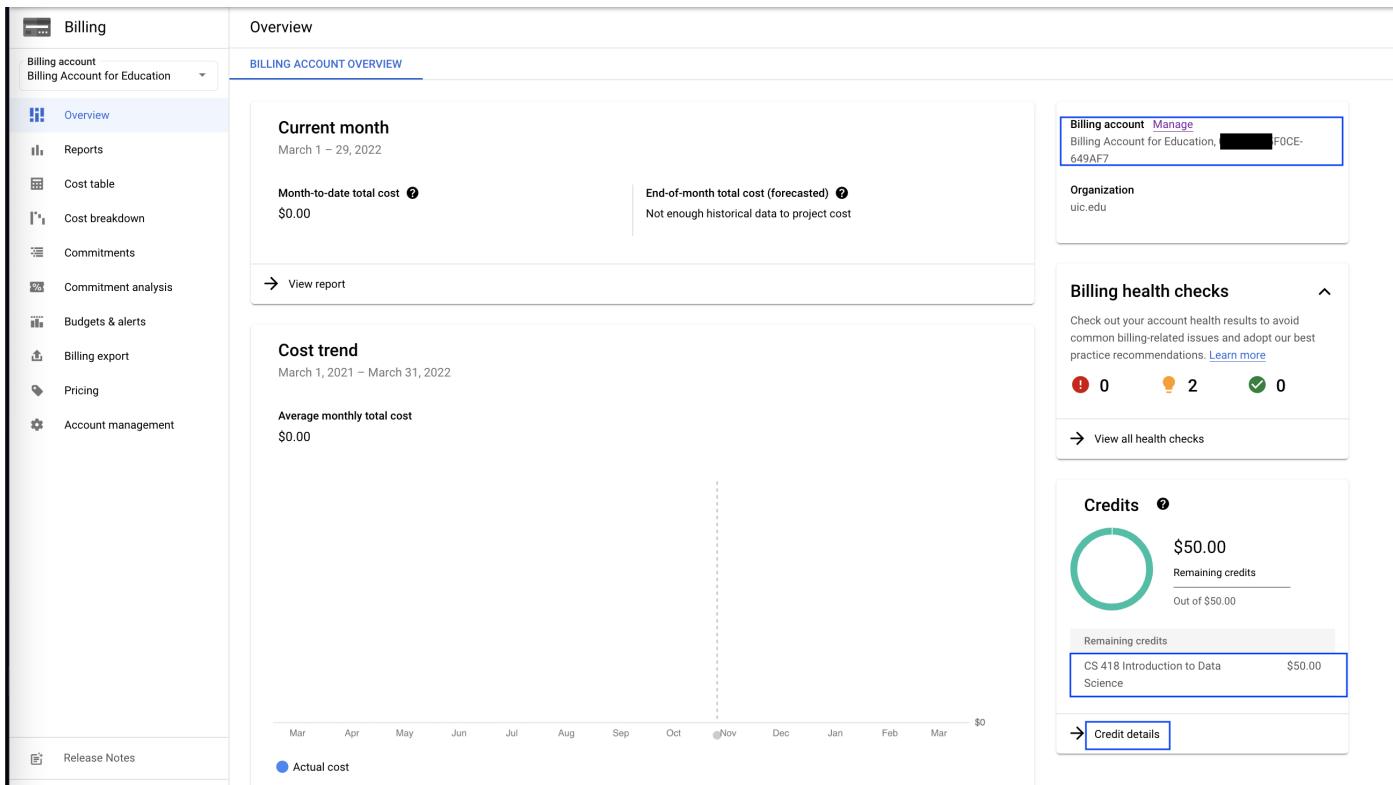
Coupon Valid Through: [1/11/2023](#)

If you have any questions, please contact your course instructor as listed above.

Thanks,

Google Cloud Education Programs Team

Once you accept and add credit to your account, you will be sent to the following page (Navigation Menu -> Billing -> Overview).

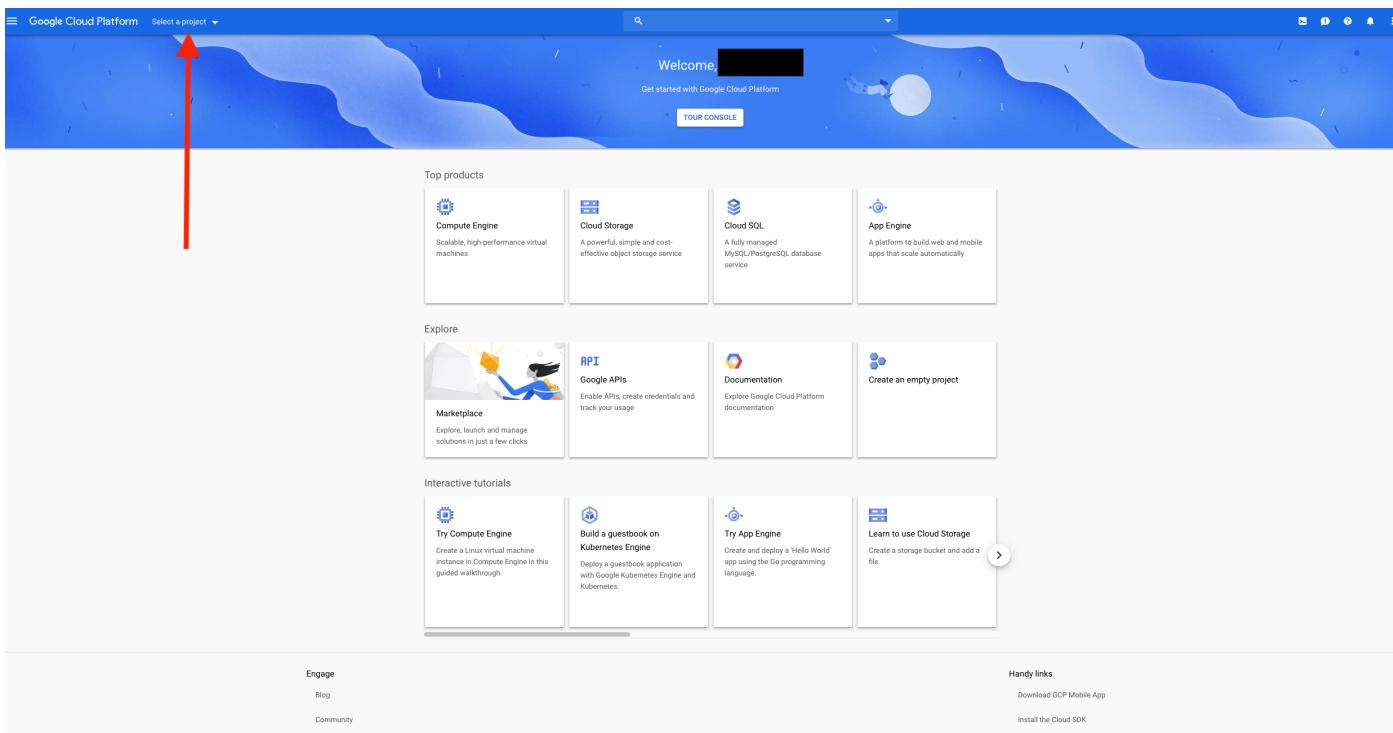


Verify that \$50 have been added to your billing account "Billing Account for Education" (default name) with credit name "CS 418 Introduction to Data Science".

Create Project

Google Cloud provides a common interface to manage and monitor all projects, this is called Google Cloud Console. First, you need to log into [Google Cloud Console \(<https://console.cloud.google.com>\)](https://console.cloud.google.com), using your university ID.

After logging in, you will land on a welcome page which, assuming you have not setup a project before, lists some popular services and tutorials shown below:



Click on the dropdown "Select a Project" indicated by the red arrow in the image above and you will be taken to this menu:

Select from NO ORGANIZATION ▾

RECENT ALL

Name	ID
▼ uic.edu	707780372120
My First Project	vivid-outcome-233006

CANCEL OPEN

Click on the New Project button as shown above and it will take you to project form:

New Project



You have 10 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name *

homework4



Project ID: homework4-237319. It cannot be changed later. [EDIT](#)

Organization

uic.edu



This project will be attached to uic.edu.

Location *

uic.edu

[BROWSE](#)

Parent organization or folder

[CREATE](#)

[CANCEL](#)

Name your project `homework4` and click *Create* button. And just as easily, your project is set up.

All the Google Cloud services are offered within the context of a project and once the project is created, you will be redirected to project dashboard (or Google Cloud Console) where you can add services to the project, monitor activity and manage billing etc.

The screenshot shows the Google Cloud Platform Home page. The sidebar on the left is open, revealing various project management and developer tools. The 'Billing' option under the 'APIs & Services' section is highlighted with a red arrow. The main content area displays several cards: 'Project info' (with project name 'homework4'), 'API APIs' (with a chart showing requests per second), 'Google Cloud Platform status' (normal), 'Billing' (estimated charges \$0.00), 'Error Reporting' (no errors), 'News' (recent articles), and 'Documentation' (links to Compute Engine, Cloud Storage, and App Engine). A red arrow points from the sidebar's 'Billing' link to the main content area's 'Billing' card.

Link Billing Account

Before we proceed, we need to link a billing account to our project or verify that the previously created billing account is linked to our project automatically. First, ensure that the correct project is selected, then open the left sidebar menu using the button on top-left and select Billing.

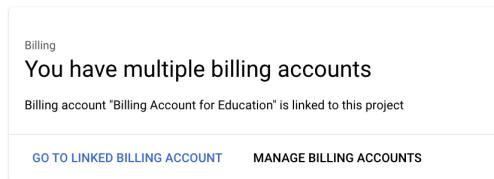
The screenshot shows the Google Cloud Platform Home page with three red annotations overlaid:

- 1. Active Project**: Points to the 'homework4' project name in the top navigation bar.
- 2. Open Sidebar Menu**: Points to the three-dot menu icon in the top-left corner of the sidebar.
- 3. Click Billing**: Points to the 'Billing' option in the sidebar menu.

The main content area displays the same cards as the previous screenshot, including the 'Billing' card which now shows 'Estimated charges USD \$0.00'.

This will take you to a screen which specifies the billing account linked to your project. Verify that the linked account is "Billing Account for Education" with \$50 credits for "CS 418 Introduction to Data Science".

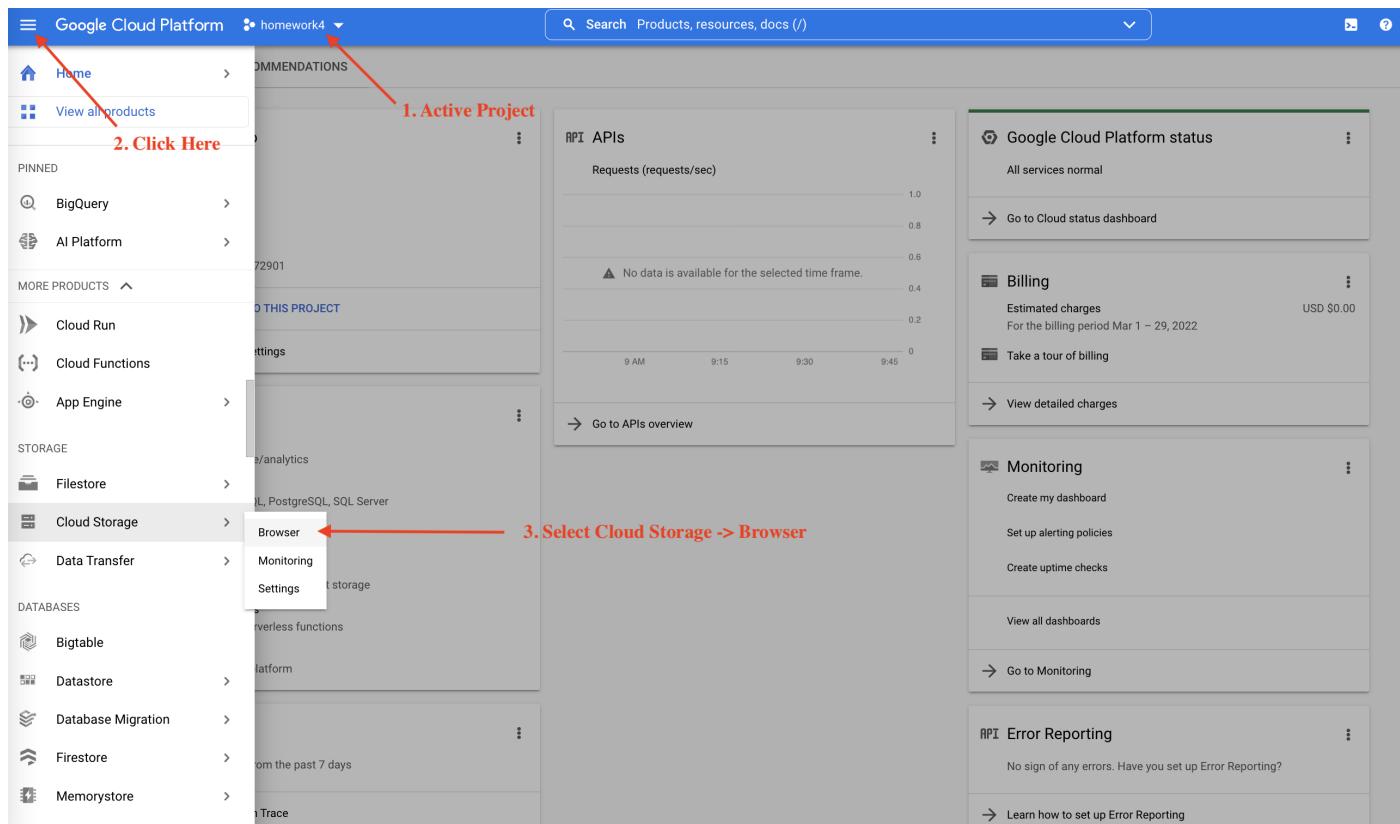
⚠ You might have multiple billing accounts, as shown in the image below, in that case, make sure that the correct account is linked to avoid any unnecessary cost.



Now that our project is setup, let's add a service to it!

Adding a Bucket

A bucket is just a storage container, it serves the same purpose as a cloud storage service (e.g. *Google Drive*) but it is more suitable for programmatic access and integration with other services as we will see later. Buckets are managed through a service *Cloud Storage* and we can add a bucket to our project through the sidebar (top-left menu) as shown in the following image:



Before you add any service, make sure that your active project is the one you just created. Then you can select the sidebar menu by clicking on the top-left button and add services from the menu.

⚠ Adding services may incur a high cost so only add what you need.

Selecting *Browser* will take you to the following screen:

Google Cloud Platform homework4

Cloud Storage

Browser

+ CREATE BUCKET

DELETE

REFRESH

HELP ASSISTANT

SHOW INFO PA

Filter buckets

Name ↑

Created

Location type

Location

Default storage class

Last modified

Public access

Access control

No rows to display

Store and retrieve your data
Get started by creating a bucket – a container where you can organize and control access to your data and files in Cloud Storage.

CREATE BUCKET

TAKE QUICKSTART

Click on the *Create Bucket* button as shown above and you will be taken to the following form:

← Create a bucket

to previous page

HELP ASSISTANT

Name your bucket

Pick a globally unique, permanent name. [Naming guidelines](#)

edu-uic-cs418-your_nick_name-homework4

Tip: Don't include any sensitive information

LABELS (OPTIONAL)

CONTINUE

1. Name in above format
Use your nick name or UIC ID to make your bucket name unique.

2. Continue

Choose where to store your data

Name your bucket

Name: edu-uic-cs418-your_nick_name-homework4

• Choose where to store your data

This permanent choice defines the geographic placement of your data and affects cost, performance, and availability. [Learn more](#)

Location type

- Multi-region
Highest availability across largest area
- Dual-region
High availability and low latency across 2 regions
- Region 3.
Lowest latency within a single region

Location

us-central1 (Iowa)

4.

CONTINUE

5.

Choose where to store your data

Location: us-central1 (Iowa)

Location type: Region

• Choose a default storage class for your data

A storage class sets costs for storage, retrieval, and operations. Pick a default storage class based on how long you plan to store your data and how often it will be accessed. [Learn more](#)

- Standard ?
Best for short term storage and frequently accessed data
- Nearline 6.
Best for backups and data accessed less than once a month
- Coldline
Best for disaster recovery and data accessed less than once a quarter
- Archive
Best for long-term digital preservation of data accessed less than once a year

CONTINUE

7.

Choose a default storage class for your data

| Default storage class: Standard

• Choose how to control access to objects

Prevent public access

Restrict data from being publicly accessible via the internet. Will prevent this bucket from being used for web hosting. [Learn more](#)

Enforce public access prevention on this bucket

Access control

Uniform

Ensure uniform access to all objects in the bucket by using only bucket-level permissions (IAM). This option becomes permanent after 90 days. [Learn more](#)

Fine-grained 8.

Specify access to individual objects by using object-level permissions (ACLs) in addition to your bucket-level permissions (IAM). [Learn more](#)

9. **CONTINUE**

• Choose how to protect object data

Your data is always protected with Cloud Storage but you can also choose from these additional data protection options to prevent data loss. Note that object versioning and retention policies cannot be used together.

Protection tools

None 10.

Object versioning (best for data recovery)

For restoring deleted or overwritten objects. To minimize the cost of storing versions, we recommend limiting the number of noncurrent versions per object and scheduling them to expire after a number of days. [Learn more](#)

Retention policy (best for compliance)

For preventing the deletion or modification of the bucket's objects for a specified minimum duration of time after being uploaded. [Learn more](#)

▼ DATA ENCRYPTION

11.

CREATE

CANCEL

Fill out the form as shown above (use the provided links to learn more about these fields), however you need to enter a unique bucket name and press create. Now you should be able to view your bucket in a file browser like environment. We will come back to this later, first, let's set up authentication first so that we can communicate with Google Cloud.

Setup Authentication

We will follow the simple steps listed in this tutorial ([Authentication tutorial](https://cloud.google.com/docs/authentication/getting-started) (<https://cloud.google.com/docs/authentication/getting-started>)) to setup authentication.

First, we need to setup a service account, open the tutorial and click on "Go To Create Service Account" button and fill up the form as shown below. When the account is created, you will be prompted to download a JSON key, **save it in homework root directory as** `homework4-key.json`.

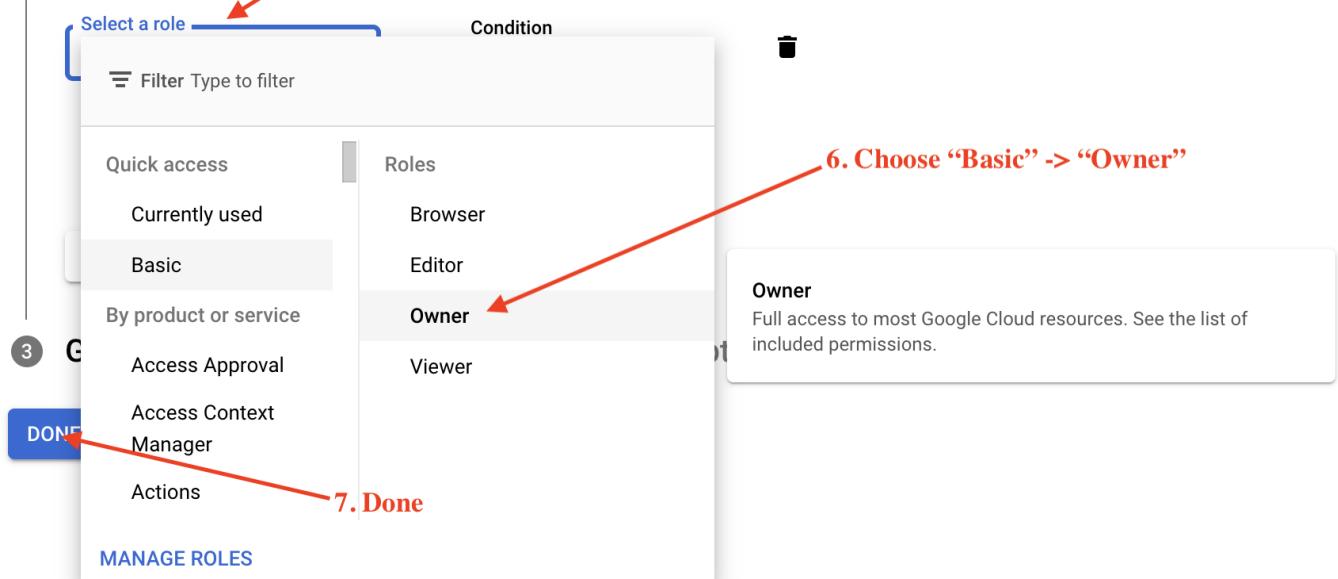
The screenshot shows the Google Cloud Platform IAM & Admin interface. On the left, a sidebar lists various services: IAM & Admin (selected), IAM, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts (selected), Workload Identity Federation, Labels, Tags, Settings, Privacy & Security, Identity-Aware Proxy, Manage Resources, and Release Notes. At the top, there is a dropdown menu for the project named "homework4" and a search bar. The main area is titled "Service account details". Step 1, "Select Project", points to the "homework4" dropdown. Step 2, "Account name", points to the "Service account name" field containing "mycs418Sp22hw4". Step 3, "Leave self-generated ID", points to the "Service account ID" field containing "mycs418sp22hw4". Step 4, "Click Here", points to the "CREATE AND CONTINUE" button. Below the main form, there are two optional sections: "Grant this service account access to project (optional)" and "Grant users access to this service account (optional)". At the bottom, there are "DONE" and "CANCEL" buttons.

Create service account

Service account details

2 Grant this service account access to project
(optional)   5. Click "Select a role"

Grant this service account access to homework4 so that it has permission to complete specific actions on the resources in your project. [Learn more](#)



Service accounts		+ CREATE SERVICE ACCOUNT	DELETE	+ MANAGE ACCESS	REFRESH	HELP ASSISTANT
Service accounts for project "homework4"						
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.						
Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.						
Email	Status	Name	Description	Key ID	Actions	
<input type="checkbox"/>					⋮	
<input type="checkbox"/>					⋮	
<input type="checkbox"/>					⋮	
<input type="checkbox"/>		mycs418Sp22hw4	mycs418Sp22hw4		⋮	

The screenshot shows the Google Cloud Platform interface for managing service account keys. On the left, a sidebar lists various IAM-related options like IAM, Identity & Organization, Policy Troubleshooter, etc. The main area shows a 'Keys' tab selected under 'mycs418Sp22hw4'. A modal window titled 'Create private key for "mycs418Sp22hw4"' is open. It contains instructions about service account keys being a security risk. Below this, there's a section to 'Add a new key pair or update existing ones'. A button labeled 'ADD KEY' is highlighted with a red arrow. The 'Key type' section shows two options: 'JSON' (selected) and 'P12'. 'JSON' is described as 'Recommended'. A red arrow points from the 'JSON' radio button to the text '11. Select "JSON"'. At the bottom of the modal, there are 'CANCEL' and 'CREATE' buttons, with a red arrow pointing to the 'CREATE' button and the text '12. Click "CREATE" to save the JSON as instructed.'

We can now setup an enviornment variable `GOOGLE_APPLICATION_CREDENTIALS` using the following python command, this enables the google cloud API to use the JSON key while making requests.

```
In [4]: os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'homework4-key.json'
```

Now, we can test our setup by making an authenticated API call to list buckets.

```
In [5]: def list_buckets():
    storage_client = storage.Client()
    # Make an authenticated API request
    buckets = list(storage_client.list_buckets())
    print(buckets)

list_buckets()
# The expected output of this command should contain the name of your bucket
# [<Bucket: [YOUR BUCKET NAME]>]
```

```
[<Bucket: edu-uic-cs418-myelyu2-homework4>]
```

Using this JSON key, we can now communicate with server without having to explicitly log in.

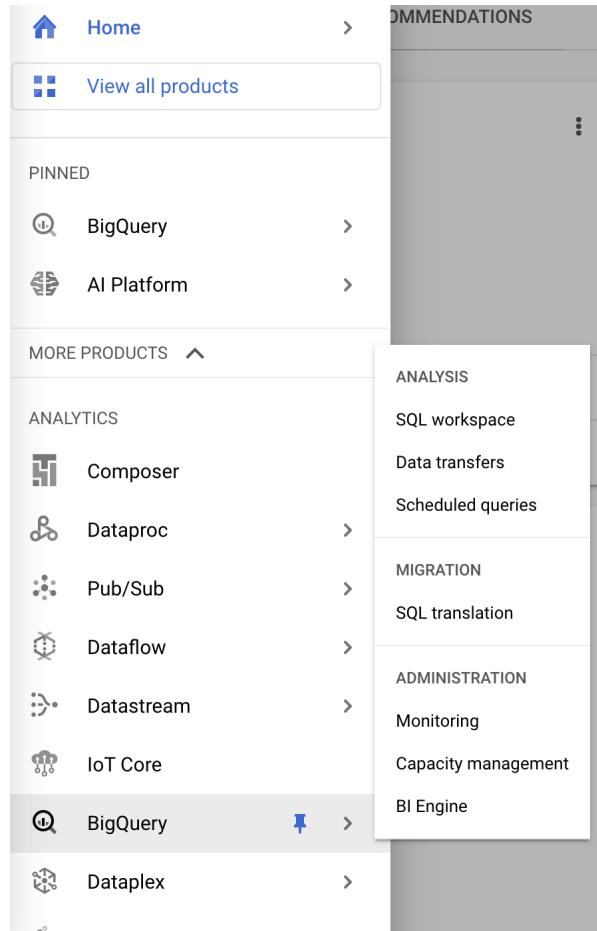
⚠ `homework4-key.json` contains sensitive information that you should not share with anyone. Do not submit this file.

Part 1: BigQuery

BigQuery is a large scale database which can process multiple GBs of data (e.g. all crime records in Chicago since 2001) within just a few seconds and provides the capacity to store and retrieve structured information which can either be used by your application or another cloud service. You can read more about it on the [project page](https://cloud.google.com/bigquery/) (<https://cloud.google.com/bigquery/>).

One of the key strengths of BigQuery is that it supports Structured Query Language (SQL) which is the de facto query language for Relational Databases in which data is stored in tabular form (just like pandas! but not in-memory). This enables the developers to query large databases without having to learn a new language, however, if you are not familiar with SQL or a simple refresher course then you can get a quick introduction at [W3 Schools](https://www.w3schools.com/sql/) (<https://www.w3schools.com/sql/>).

Let's first go to BigQuery interface and explore some large scale public datasets. From the [project dashboard](https://console.cloud.google.com) (<https://console.cloud.google.com>) select BigQuery from the sidebar:



This will take you to the BigQuery interface. For this homework, we will be using publicly available chicago crime data which encompasses crime records in Chicago since 2001. It is already structured, cleaned and maintained by Google and we can just directly query it, however we can also add our own datasets to BigQuery if needed.

SANDBOX Set up billing to upgrade to the full BigQuery experience

Marketplace

Marketplace > "chicago crime data" > Datasets

Datasets

1. ADD DATA -> Explore public

2. Search

3. Click here and click "View Dataset" in the next window

4. Public-data is pinned in new tab

5. Explore required dataset

Chicago Crime Data
City of Chicago

This dataset reflects reported incidents of crime (with the exception of Chicago from 2001 to present, minus the most recent seven days) from the Chicago Police Department's (Enforcement Analysis and Reporting) system. In order to protect

Browse and select the table from the sidebar as shown in the figure and you will be able to see all the fields in the table and their description. You can also select *Preview* to show a subset of the dataset.

Google Cloud Platform

homework4

SEARCH Products, resources, docs (/)

FEATURES & INFO SHORTCUT DISABLE EDITOR TABS

Explorer + ADD DATA

Type to search

Public Datasets

Looking for something? We've moved all datasets to the new public datasets project, and public access. Expand projects to see them.

Editor * UNSAVE... COMPOSE NEW QUERY

RUN This query will process 1.47 GB when run.

```
1 SELECT * FROM `bigquery-public-data.chicago_crime.crime`
2 | LIMIT 1000
```

CRIME

crime

QUERY SHARE COPY

SCHEMA DETAILS PREVIEW

Table schema

Processing location: US

Query results

RESULTS JSON EXECUTION DETAILS

Row	unique_key	case_number	date	block
1	7688495	HS494701	2010-09-02 12:18:00 UTC	069XX W DIVER
2	7733365	HS539417	2010-09-29 08:45:00 UTC	049XX S ARCHE
3	7733875	HS541204	2010-09-26 11:30:00 UTC	051XX S WENTW
4	872879	HT103804	2011-01-03 04:55:00 UTC	062XX S NATOM

PERSONAL HISTORY PROJECT HISTORY SAVED QUERIES

Selected Table

Results per page: 50 1 – 50 of 1000

Let's run this simple query using the Query Editor

```
#standardSQL
```

```
SELECT * FROM `bigquery-public-data.chicago_crime.crime`
WHERE primary_type = "THEFT"
LIMIT 100;
```

This query will select the top 100 crime records of type theft from the crime dataset.

⚠ ` or backtick is a character which is not the same as an apostrophe or double quotes and is often used in SQL.

⚠ You always need to enter the prefix of the table or the dataset identifier `bigquery-public-data.chicago_crime` to access `crime` table.

Query editor HIDE EDITOR

```
1 SELECT * FROM `bigquery-public-data.chicago_crime.crime`
2 WHERE primary_type = "THEFT"
3 LIMIT 100;
```



Query results [SAVE RESULTS](#) [EXPLORE IN DATA STUDIO](#)

Query complete (1.4 sec elapsed, 1.3 GB processed)

Job information	Results	JSON	Execution details												
Row	unique_key	case_number	date	block	iucr	primary_type	description	location_description	arrest	domestic	beat	district	ward	community_area	fbi_code
1	7817711	HS627068	2010-11-22 07:35:00 UTC	047XX S WENTWORTH AVE	0850	THEFT	ATTEMPT THEFT	CTA TRAIN	false	false	231	2	3	37	06
2	5543327	HN343432	2006-11-20 09:00:00 UTC	062XX N LEONA AVE	0841	THEFT	FINANCIAL ID THEFT:\$300 &UNDER	RESIDENCE	false	false	1621	16	45	12	06
3	5579230	HN387320	2007-06-05 20:49:48 UTC	007XX E 38TH ST	0850	THEFT	ATTEMPT THEFT	STREET	true	false	212	2	4	36	06
...

In the figure above, you can see the results listed as a table. The Query editor shows you how much data it is going to process when the query runs so that you may estimate the cost. In this case, we are running this query on 1.3 GB of data within a few seconds. You may also export results in CSV format or explore them in Data Studio (a visualization portal that we will discuss later). Just like `pandas`, you may also perform a Group By operation and order the results using the following query:

```
SELECT primary_type, COUNT(*) as count FROM `bigquery-public-data.chicago_crime.crime`
GROUP BY primary_type
ORDER BY count;
```

The results are as follows:

The screenshot shows a BigQuery query interface. At the top, there's a toolbar with 'EDITOR', 'RUN' (highlighted in blue), 'COMPOSE NEW QUERY', and a note that the query will process 85.99 MB when run. Below the toolbar is the SQL query:

```

1 SELECT primary_type, COUNT(*) as count
2 FROM `bigquery-public-data.chicago_crime.crime`
3 GROUP BY primary_type
4 ORDER BY count;

```

Below the query is a note: 'Processing location: US'. The main area is titled 'Query results' with tabs for 'SAVE RESULTS' and 'EXPLORE DATA'. There are three tabs at the top of the results table: 'JOB INFORMATION', 'RESULTS' (selected), and 'JSON'. The results table has columns: Row, primary_type, and count. The data is as follows:

Row	primary_type	count
1	DOMESTIC VIOLENCE	1
2	NON-CRIMINAL (SUBJECT SPECIFIED)	9
3	RITUALISM	24
4	NON - CRIMINAL	38
5	HUMAN TRAFFICKING	89
6	OTHER NARCOTIC VIOLATION	141
7	NON-CRIMINAL	176
8	PUBLIC INDECENCY	188
9	OBSCENITY	760
10	CONCEALED CARRY LICENSE VIOLATION	882

At the bottom, it says 'Results per page: 100 ▾ 1 – 36 of 36' with navigation arrows.

Q1 (20%)

✍ Write an SQL query to extract the number of arrests/no-arrests per primary type for domestic-related cases, exclude "OTHER OFFENSE" and all non-criminal types, only consider records until the end of year 2021 and sort the results in ascending order by primary type and arrest.

💻 Take a screenshot (full screen, png format), rename it as `q1-results.png` and store it in `sub` subdirectory.

✍ Include screenshot in the notebook by editing the following markdown (if needed). It shows as a broken image by default if screenshot is unavailable.

The screenshot shows the Google Cloud Platform BigQuery interface. On the left, the sidebar lists various datasets and tables, with 'crime' selected. The main area shows the schema for the 'crime' table, which includes fields like unique_key, case_number, date, block, iucr, primary_type, description, location_description, arrest, domestic, beat, district, ward, and community_area. Below the schema, there are tabs for PERSONAL HISTORY, PROJECT HISTORY, and SAVED QUERIES. To the right, a query editor tab is open with the following SQL code:

```

1 SELECT primary_type, arrest, COUNT(*) as counts
2 FROM `bigquery-public-data.chicago_crime.crime`
3 WHERE year <= 2021
4 AND domestic = TRUE
5 AND primary_type NOT LIKE '%OTHER OFFENSE%'
6 AND primary_type NOT LIKE '%NON-CRIMINAL%'
7 GROUP BY primary_type, arrest
8 ORDER BY primary_type, arrest;

```

The results table shows the count of crimes for each primary type. The results per page are set to 50, and there are 54 pages.

⚠️ Refresh your browser if it does not appear right away.

Since pandas and SQL operate on tables and offer similar functionality, we can transfer SQL results to a pandas data frame for further processing. One arduous way to do it is to first export results from BigQuery interface as a CSV file and load it in pandas, but there is a much better and convenient method available through the `bigrquery` magic command. The following statement will run a query on the server and store results in the specified `df` dataframe, while running it shows the time elapsed.

In [6]: `%%bigrquery df`

-- This is an SQL comment

```
SELECT * FROM `bigquery-public-data.chicago_crime.crime`
WHERE primary_type = "THEFT"
LIMIT 100;
```

Query complete after 0.04s: 100%

1/1 [00:00<00:00, 262.88query/s]

Downloading: 100%

00/100 [00:01<00:00, 66.70rows/s]

```
In [7]: # We can now print a subset of these results using this familiar method  
df.head()
```

Out[7]:

	unique_key	case_number	date	block	iucr	primary_type	description	location_description	arrest
0	4545973	HM132340	2006-01-19 02:50:00+00:00	066XX W HAYES AVE	0850	THEFT	ATTEMPT THEFT	VEHICLE NON-COMMERCIAL	Fa
1	8044404	HT276493	2011-05-02 05:00:00+00:00	022XX S ARCHER AVE	0880	THEFT	PURSE-SNATCHING	SIDEWALK	Fa
2	8137142	HT371226	2011-06-29 05:50:00+00:00	003XX W 47TH ST	0880	THEFT	PURSE-SNATCHING	SIDEWALK	Fa
3	8216681	HT450703	2011-08-15 09:00:00+00:00	088XX S PAXTON AVE	0841	THEFT	FINANCIAL ID THEFT:\$300 &UNDER	RESIDENCE	Fa
4	3113298	HJ846122	2003-12-30 04:45:00+00:00	091XX S DREXEL AVE	0880	THEFT	PURSE-SNATCHING	SIDEWALK	Fa

Copy the query you wrote earlier to retrieve the number of arrests per each primary type for domestic-related cases in the following cell.

⚠️ This is important!

```
In [8]: %%bqqlfql df
```

```
-- Copy your SQL query below this line. (Make sure you rename the count aggregation

SELECT primary_type, arrest, COUNT(*) as counts
FROM `bigquery-public-data.chicago_crime.crime`
WHERE year <= 2021
    AND domestic = TRUE
    AND primary_type NOT LIKE '%OTHER OFFENSE%'
    AND primary_type NOT LIKE '%NON-CRIMINAL%'
GROUP BY primary_type, arrest
ORDER BY primary_type, arrest;
```

Query complete after 0.00s: 100%|██████████

1/1 [00:00<00:00, 572.76query/s]

Downloading: 100%|██████████

54/54 [00:01<00:00, 44.16rows/s]

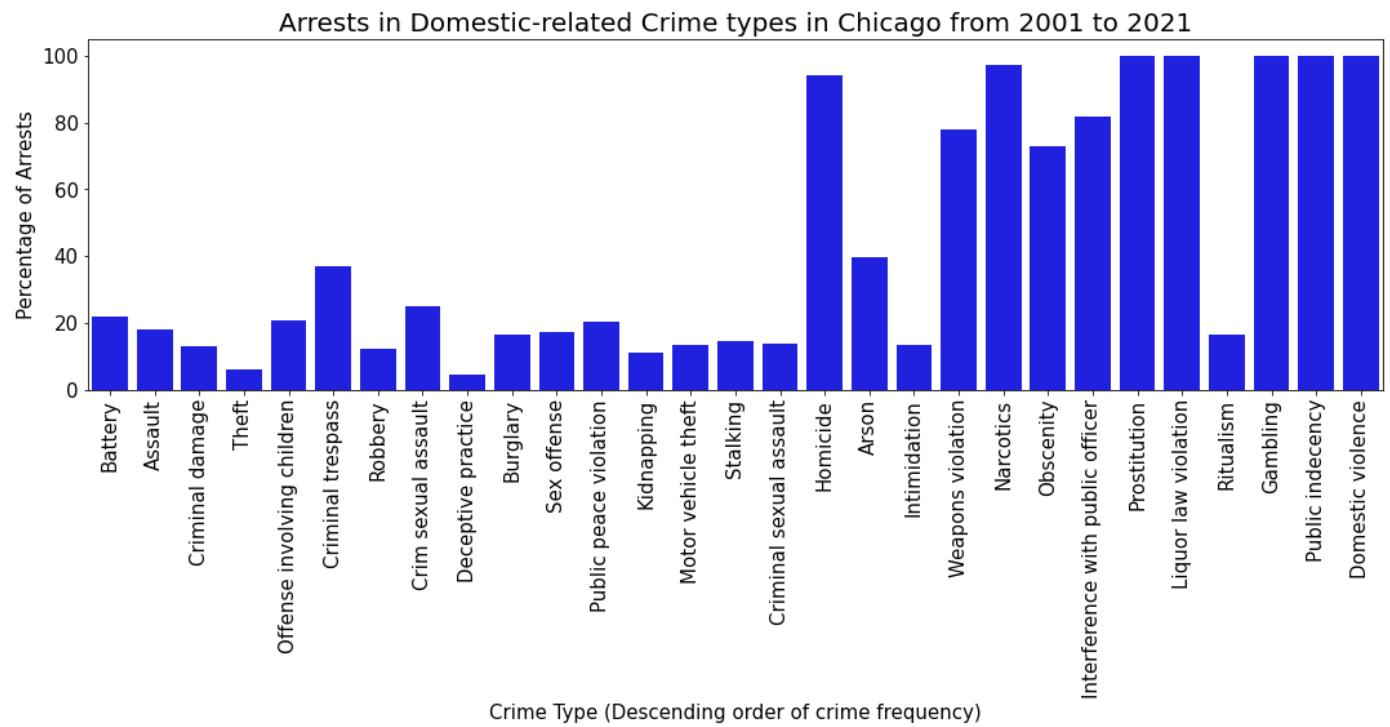
```
In [9]: # We can now view a sample of our results
df = qldf.copy()
qldf.head()
# primary_type  arrest  counts
# 0 ARSON    False   417
# 1 ARSON    True    275
# 2 ASSAULT  False  88889
# 3 ASSAULT  True   19855
# 4 BATTERY  False  463715
```

Out[9]:

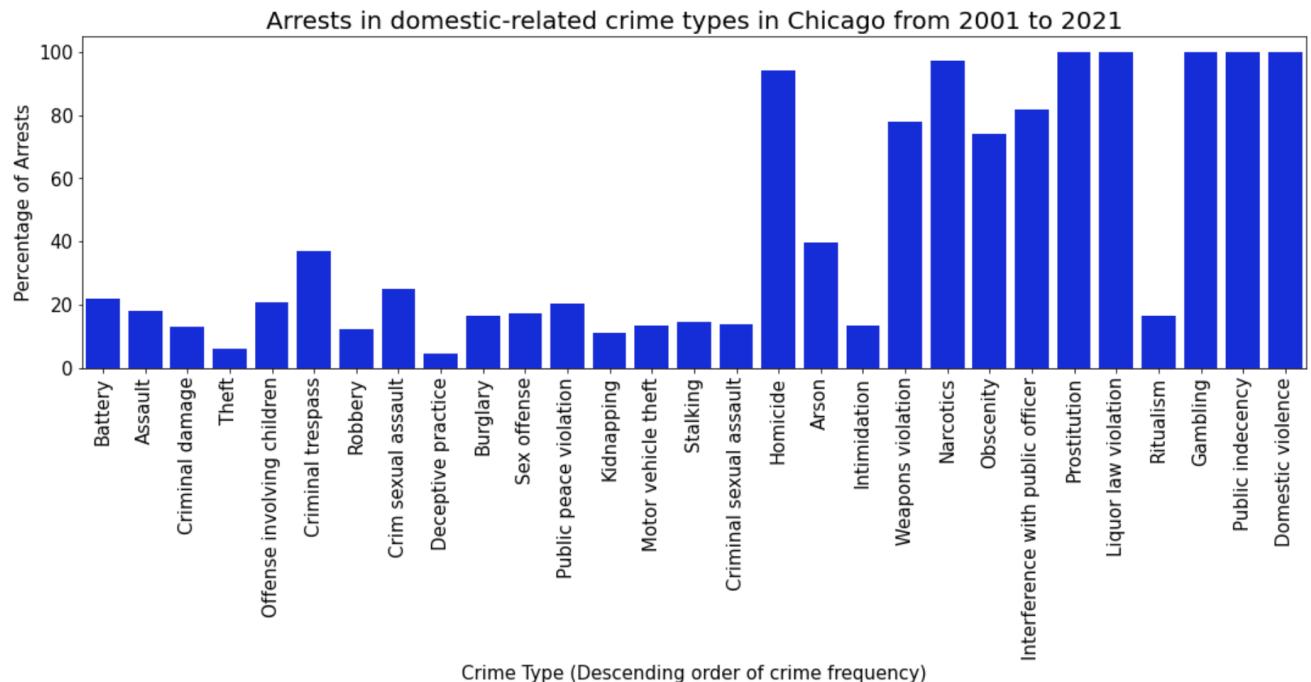
	primary_type	arrest	counts
0	ARSON	False	417
1	ARSON	True	275
2	ASSAULT	False	88889
3	ASSAULT	True	19856
4	BATTERY	False	463714

```
In [10]: # Using the same dataframe, we can create a plot showing the fraction of arrests for
fig = plt.figure(figsize=(18, 5))
```

```
# Normalizing results
qldf["crime_freq"] = qldf.groupby("primary_type")["counts"].transform(lambda x:x.sum())
qldf["percentage"] = qldf["counts"].mul(100.0).divide(qldf["crime_freq"])
qldf["primary_type"] = qldf["primary_type"].apply(lambda x: x.capitalize())
norm_df = qldf[qldf["arrest"] == True]
norm_df = norm_df.sort_values(by="crime_freq", ascending=False)
# Standard sns barplot
ax = sns.barplot(x="primary_type", y="percentage", data=norm_df, color="b")
# Adding context to the plot
plt.xticks(rotation=90, fontsize=15)
plt.yticks(fontsize=15)
plt.xlabel("Crime Type (Descending order of crime frequency)", fontsize=15)
plt.ylabel("Percentage of Arrests", fontsize=15)
plt.title("Arrests in Domestic-related Crime types in Chicago from 2001 to 2021", fontcolor="red")
plt.show()
```



Your graph should look like the graph below:



```
In [11]: # Now we can save this file in our submission subdirectory
# Note you are submitting results of your original query not processed dataframe
out_filepath = os.path.join("sub", "crime_arrests.csv")
df.to_csv(out_filepath, index=False)
```

Check your `sub` (submission) subdirectory to verify that you have a file named `crime_arrests.csv`.

Q2 (20%)

For each location category (column `location_description`) find the most frequent crime type. Your final results (in the CSV) should have three columns `location_description`, `primary_type`, `counts`. Where `location_description` is the category for different locations such as streets, residence, etc., `primary_type` is the most frequent crime on that location category and `counts` is simply the number of records associated with that crime. Exclude "OTHER OFFENSE", all non-criminal types and null values of `location_description`. Only consider records until the end of year 2021 and sort results in ascending order by `location_description`.

It would be simpler to first extract relevant counts using SQL and then find the most frequent one for each block using `pandas` but some SQL experts might be able to do it in a single SQL query.

```
In [12]: %%bq query q2df
```

```
#standardSQL
-- Write your query below this line

SELECT location_description, primary_type, arrest, COUNT(*) AS counts
FROM `bigquery-public-data.chicago_crime.crime`
WHERE year <= 2021
    AND primary_type NOT LIKE '%OTHER OFFENSE%'
    AND primary_type NOT LIKE '%NON-CRIMINAL%'
    AND location_description IS NOT NULL
GROUP BY primary_type, arrest, location_description
ORDER BY location_description;
```

```
Query complete after 0.00s: 100%|██████████
```

```
1/1 [00:00<00:00, 537.32query/s]
```

```
Downloading: 100%|██████████
```

```
4869 [00:01<00:00, 4150.23rows/s]
```

```
In [13]: # Write python code here for post-processing with pandas
```

```
_loc, _types = q2df['location_description'], q2df['primary_type']
unique_locations, unique_types = _loc.unique(), _types.unique()
newdf = pd.DataFrame(columns=['location_description', 'primary_type', 'counts'])

for i in range(len(unique_locations)):
    val, typ = -999, None
    for j in range(len(unique_types)):
        locs = q2df.loc[q2df['location_description'] == unique_locations[i]]
        types = locs.loc[locs['primary_type'] == unique_types[j]]
        counts = types['counts'].sum()
        if val < counts:
            val = counts
            typ = unique_types[j]
    newdf = newdf.append({'location_description': unique_locations[i], 'primary_type': typ, 'counts': val})
newdf = newdf.sort_values('location_description')
q2df = newdf.copy()
```

```
In [14]: # Let us see a subset of results  
q2df.head(10)
```

Out[14]:

	location_description	primary_type	counts
0	ABANDONED BUILDING	NARCOTICS	2985
1	AIRCRAFT	THEFT	357
2	AIRPORT BUILDING NON-TERMINAL - NON-SECURE AREA	THEFT	478
3	AIRPORT BUILDING NON-TERMINAL - SECURE AREA	THEFT	457
4	AIRPORT EXTERIOR - NON-SECURE AREA	THEFT	224
5	AIRPORT EXTERIOR - SECURE AREA	THEFT	168
6	AIRPORT PARKING LOT	THEFT	321
7	AIRPORT TERMINAL LOWER LEVEL - NON-SECURE AREA	CRIMINAL TRESPASS	865
8	AIRPORT TERMINAL LOWER LEVEL - SECURE AREA	THEFT	546
9	AIRPORT TERMINAL MEZZANINE - NON-SECURE AREA	THEFT	49

```
In [15]: # Now we can save this file in our submission subdirectory  
out_filepath = os.path.join("sub", "freq_crime.csv")  
q2df.to_csv(out_filepath, index=False)
```

Check your sub (submission) subdirectory to verify that you have a file named freq_crime.csv following the format specified above.

Part 2: Google Data Studio

In this part, we will explore a tool called [Google Data Studio](https://datastudio.google.com) (<https://datastudio.google.com>) which allows us to create visualizations using large scale datasets. This tool can retrieve data from many sources (including BigQuery tables!) and provide a simple interface to create dynamic and sophisticated visualizations. Let's get started!

Q3 (20%)

Credit will be awarded based on completion of this tutorial. We will try to replicate the graph we created earlier in **Q1**.

First, we need to create **a new (blank) report**, use your last name as the name of report with a suffix -q3 , if your are working in group then the report name should be of the format lastname1-lastname2-q3 .

You will be directed to the following screen after report is created. Select BigQuery as the data source.

The screenshot shows the Data Studio interface with the title bar "File View Page Help" and various toolbar icons. Below the toolbar, there's a search bar and a section titled "Add data to report". Under "Connect to data", there are three main options: "BigQuery" (selected), "File Upload", and "Campaign Manager". Each option has a brief description and a "More" button.

- BigQuery: Connect to BigQuery tables and custom queries.
- File Upload: Connect to CSV (comma-separated values) files.
- Campaign Manager: Connect to Campaign Manager data.

Select the crime table from chicago_crime public dataset and click *Add* button.

This screenshot shows the "Add data to report" interface for BigQuery. It includes a header with "Untitled Report" and navigation links. Below the header is a message about the BigQuery BI Engine. The main area displays a table of datasets and tables:

	Billing Project	Public Dataset	Table
MY PROJECTS			
SHARED PROJECTS	homework4	chicago_crime	crime
CUSTOM QUERY		chicago_taxi_trips	
PUBLIC DATASETS		cloud_storage_geo_index	
		cms_codes	

At the bottom right are "Cancel" and "Add" buttons.

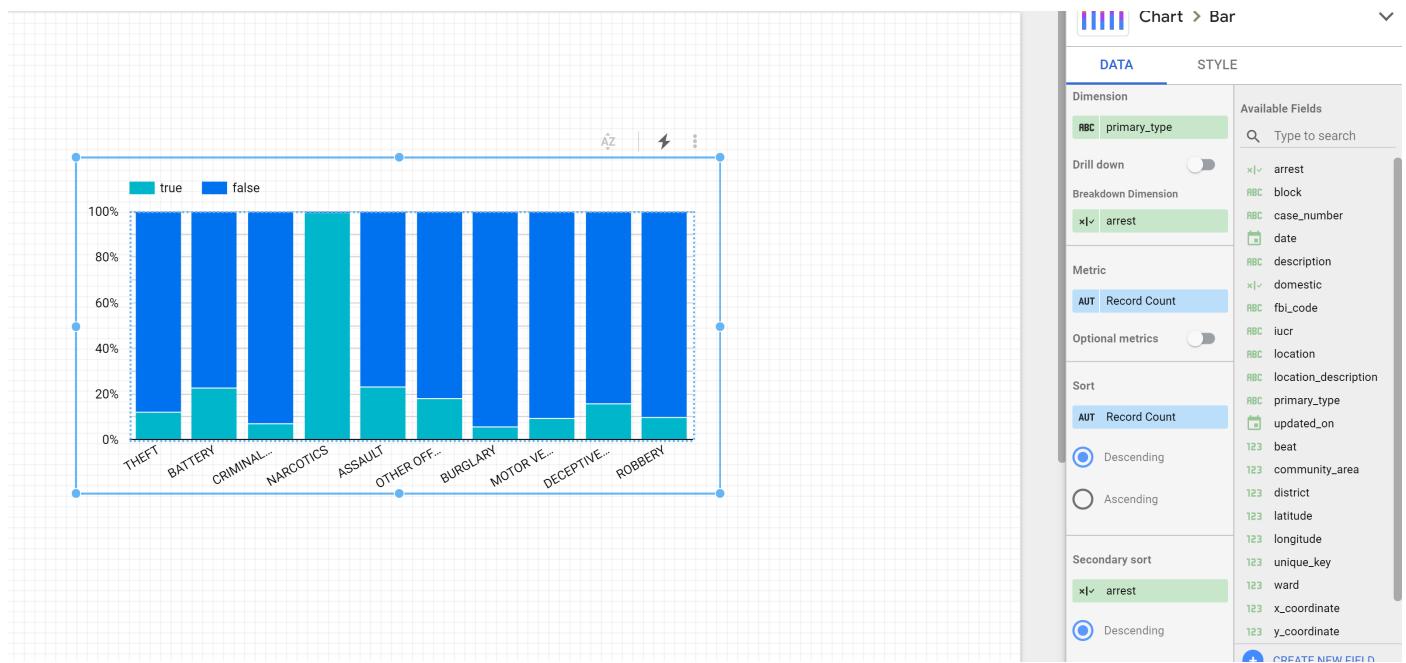
This will take you to add data interface with a default chart. Delete the chart and rename the report as instructed earlier.

The screenshot shows the Tableau desktop application. On the left, a 'Table' view displays a list of crime cases with fields 'case_number' and 'Record Count'. The data includes entries like HZ140230 (Record Count 6), HJ590004 (Record Count 6), HP26582 (Record Count 5), JC470284 (Record Count 5), HS266531 (Record Count 5), HS496074 (Record Count 4), HN217726 (Record Count 4), HU104730 (Record Count 4), and HY346207 (Record Count 4). A pagination bar at the bottom shows '1 - 100 / 7097671' with navigation arrows. On the right, the 'Chart > Table' sidebar is open, showing the 'DATA' tab selected. Under 'Data source', 'crime' is listed. Under 'Available Fields', various fields are listed: arrest, block, case_number, date, description, domestic, fbi_code, iucr, location, location_description, primary_type, updated_on, beat, and community_area. A 'Metric' section shows 'Record Count' selected. A 'STYLE' tab is also present.

Now that we have a data source added to the report, we can use it to create charts. Click *Add a chart* dropdown menu and select *100% stacked column chart* from it as shown below.

The screenshot shows the Tableau desktop application again. The 'Add a chart' dropdown is open on the top toolbar, and the '100% stacked column chart' option is highlighted with a red box. To the right, the 'Theme and Layout' sidebar is open, showing the 'THEME' tab selected. It displays a preview of a dashboard with the title 'Default' and a 'Text' card. Below the preview, there are buttons for 'Customize' and 'Extract theme from image'.

Once, the chart is added, you can click on the chart to view Chart configurations in the right sidebar. Under *DATA*, we can configure fields and under *STYLE*, we can configure appearances of the chart. Setup the fields as shown in the figure below:



Here, *Dimension* is the variable on the x-axis, *Breakdown Dimension* is the same as *hue* in seaborn. *Metric* specifies how you are going to aggregate records and *Sort* is just sorting on the x-axis. Something seems missing in this graph (can you spot it?).

Now, we need to use correct filters. Click on *ADD A FILTER* under *DATA* in the right sidebar and setup the appropriate filters as shown below:

Create Filter

Name: crime

Include: domestic True

Exclude: primary_type Equal to (=) OTHER OFFENSE

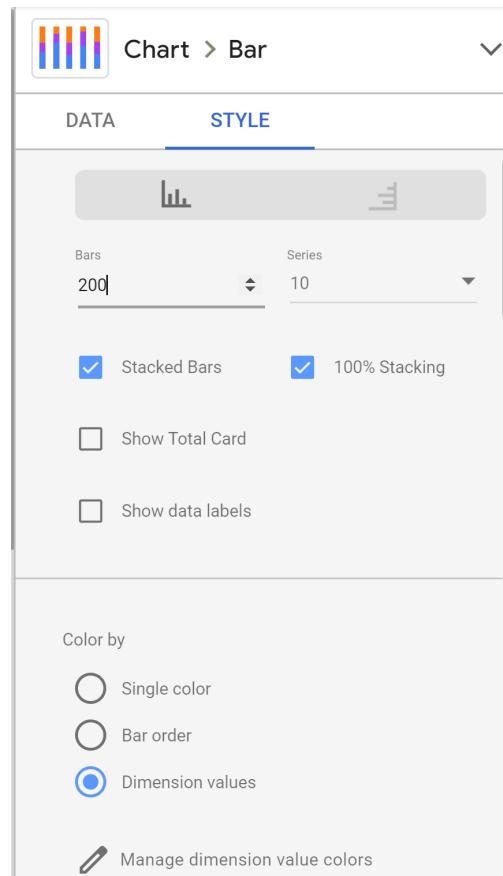
Exclude: primary_type Starts with NON

Include: year Less than (<) 2022

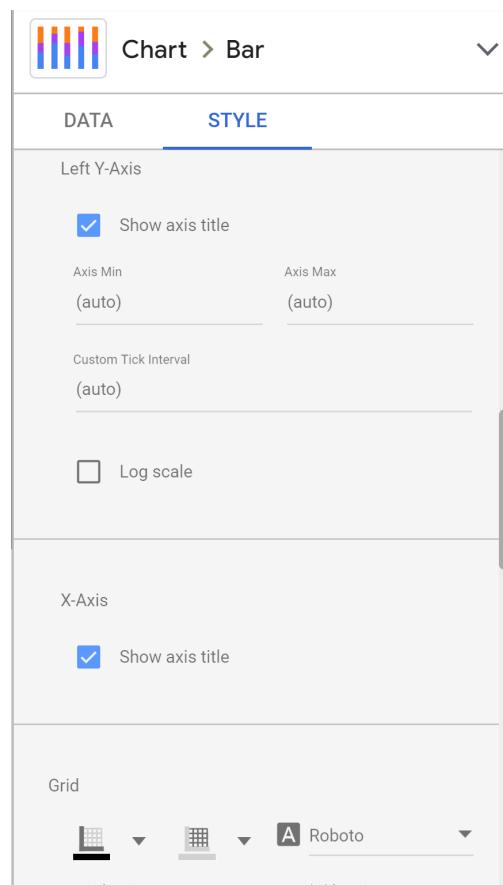
This filter has 4 clauses

SAVE

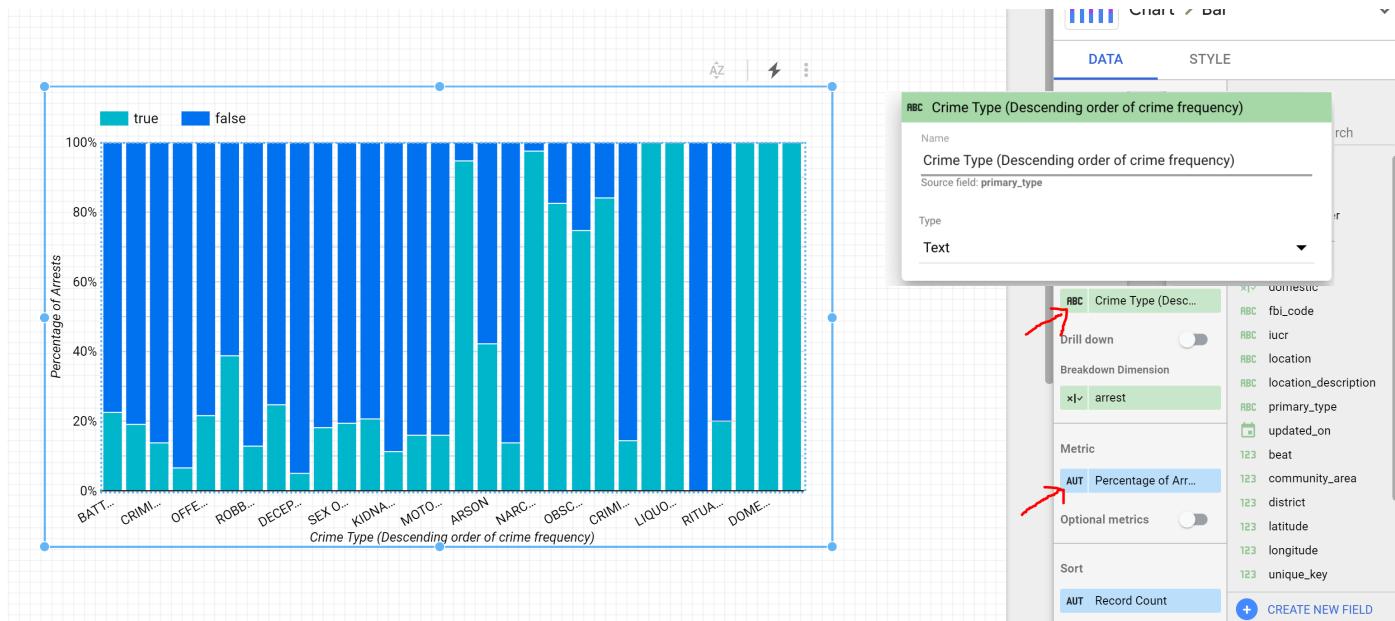
The graph we created earlier had a lot more bars than this one and this is because, by default, it only shows a subset of data. Let us fix that under the *STYLE* menu by setting *Bars* to a large number (200).



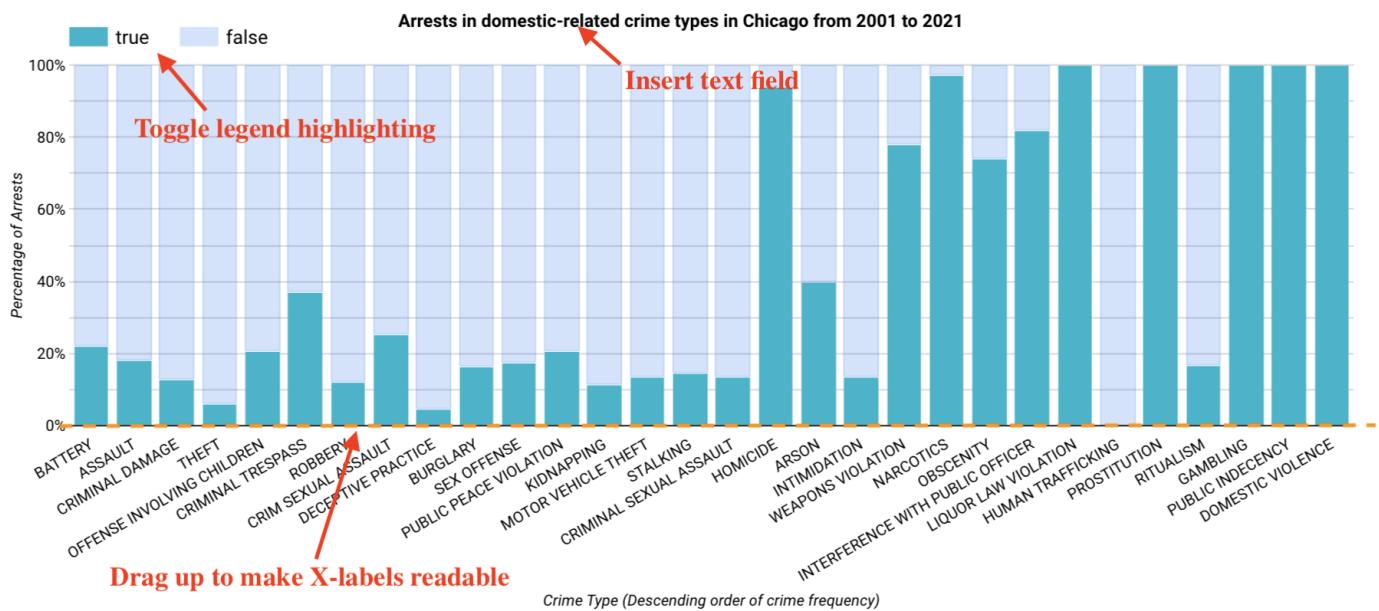
That should fix the bars, but we also want to label our axes. We can also do that in the *STYLE* menu by checking these boxes:



Now we can see all crime types and the axis labels but the labels are not very readable, we can change them by clicking the relevant areas highlighted by red arrows in the following figure.



Our graph is now almost ready, we can add title using *Text* tool from the toolbar and we can drag inner boundary (signified by dashed orange line in the figure below) to make our x-axis labels more visible. We can drag the external boundaries to stretch the graph and voila, with just a few clicks, we managed to create this graph from a large dataset.



Submission

Take a screenshot (full screen, PNG format), rename it as `q3-tutorial.png` and store it in `sub` subdirectory.

Make sure that the report name and graph is visible in the screenshot, no credit will be awarded otherwise.

Include screenshot in the notebook by editing the following markdown (if needed). It shows as a broken image by default if screenshot is unavailable.



Add page

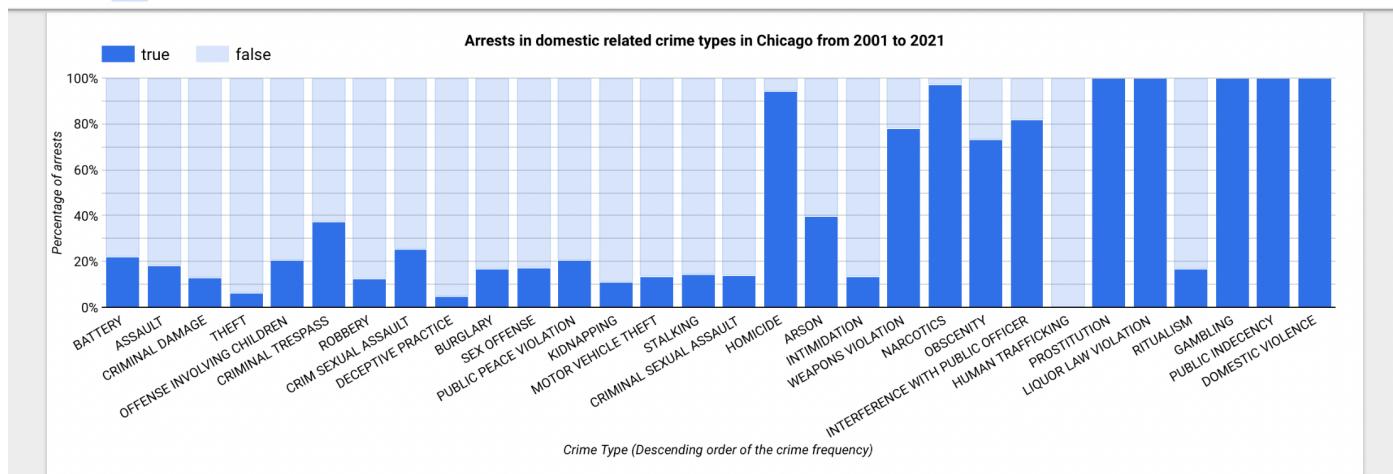
Add data

Add a chart

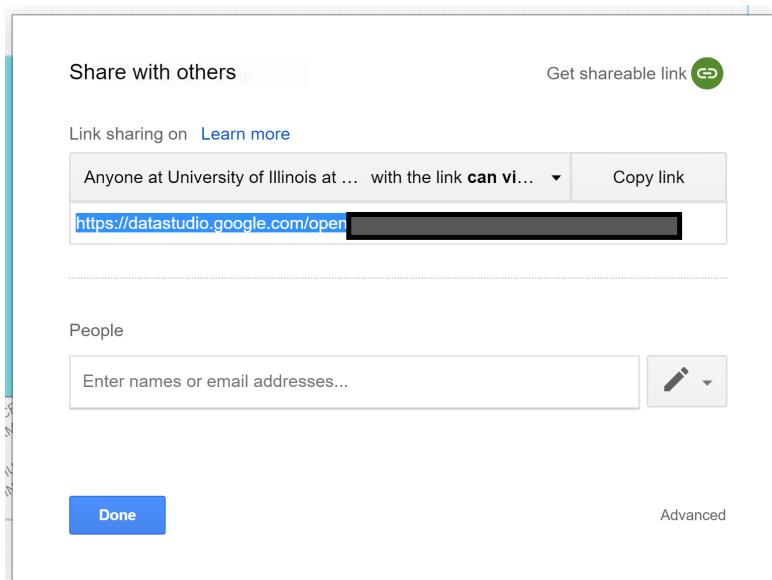
Add a control



Theme and layout



The report is now ready to be shared. Click the Share icon and get a Shareable link similar to Google docs:



⚠ Copy the sharable link to report from the previous step into the box below. Anyone within the University of Illinois should be able to view this report without having to explicitly request access.

⚠ No credit will be awarded without this link. Test this link in incognito mode of the browser.

⚠ DO NOT change/delete the report after submission.

<https://datastudio.google.com/reporting/408bddc5-6404-4e5c-ba5d-13f2c3bf146b>
(<https://datastudio.google.com/reporting/408bddc5-6404-4e5c-ba5d-13f2c3bf146b>)

Q4 (20%)

This question is more open ended and similar to Q2.2-Q2.4 from HW2. It will be graded in the same way, based on the completeness of the report produced and the insights you gained from it. We will be using the publicly available chicago crimes dataset for this task and you are only required to produce a single graph.

Be sure to consider transformations, subsets, correlations, reference markers, and lines/curves-of-best-fit (as covered in Chapter 6 of PTDS) to reveal the relationship that you are wanting to learn more about. Also be sure to make plots that are appropriate for the variable types. For completeness, be explicit about any assumptions you make in your analysis. An exemplary plot will have:

- A title
- Labelled and appropriately scaled axes
- A legend, if applicable
- A carefully selected color scheme
- A main point, accentuated through design choices

First, same as **Q3**, you need to create a new (blank) report, use your last name as the name of report with a suffix `-q4` , if your are working in group then the report name should be of the format `lastname1-lastname2-q4` .

 No credit will be awarded if the report name is wrong.

 Do not use the same report that you used in **Q3**

Submission

 Write your main takeaway/hypothesis 5-15 words in the following cell:

I want to analyze the tendency of a sex offenses in the city of Chicago. The main hypothesis is that the number of arrested people for this certain offense will increase throughout the years.

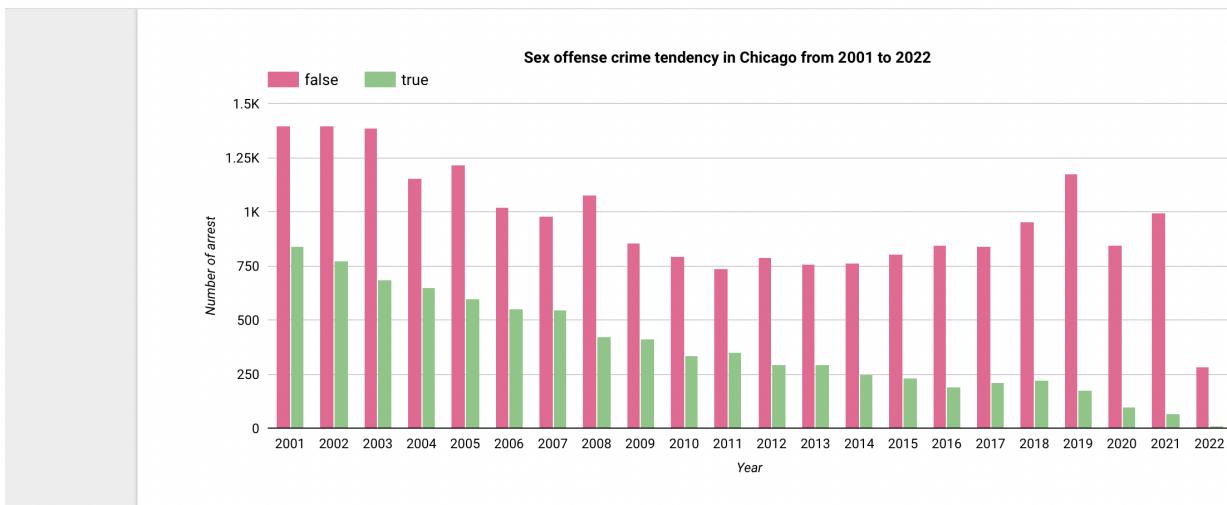
 Write a description 100-150 words following cell explaining your assumptions and what you have found.

I wanted to see how well the police of Chicago work with crimes related to a sex offense and discovered that throughout all years (2001-2022) the number of arrested people was significantly lower than those who were not arrested. The tendency of arrests is dropping every year while the number of crimes stays the same. The worst numbers were shown in 2022 (it might be because the data is incomplete) but still, in the year 2021, the number of arrested people for sex offences was equal to 69 while unarrested 995 which is very upsetting.

 Take a screenshot (full screen, PNG format), rename it as `q4-report.png` and store it in `sub` subdirectory.

 Make sure that the report name and graph is visible in the screenshot, no credit will be awarded otherwise.

 Include screenshot in the notebook by editing the following markdown (if needed). It shows as a broken image by default if screenshot is unavailable.



>Create a sharable link to this report using the process shown in **Q3** and add it in the next cell. Anyone within the university should be able to view this report without having to explicitly request access.

No credit will be awarded without this link. Test this link in incognito mode of the browser.

DO NOT change/delete the report after submission.

<https://datastudio.google.com/reporting/85f971cb-b509-45db-bd8f-c68e095c61ed>
[\(https://datastudio.google.com/reporting/85f971cb-b509-45db-bd8f-c68e095c61ed\)](https://datastudio.google.com/reporting/85f971cb-b509-45db-bd8f-c68e095c61ed)

Part 3: Google Cloud AI Platform

In this part, we will use [Google Cloud AI Platform \(https://cloud.google.com/ai-platform/docs\)](https://cloud.google.com/ai-platform/docs) to train and deploy a simple machine learning model. Training models from large datasets is a particularly tedious task, especially if it involves hyper-parameter tuning and training may take days to complete and require high computational powers and multiple CPUs/GPUs/TPUs. Google Cloud AI Platform addresses all these problems. Let's get started!

Q5 (20%)

Credit will be awarded based on completion of this tutorial. Same, as before, we will train our model using tweets dataset from *HW3* and make predictions on the test set, however we will do all that using Google Cloud AI Platform.

Twitter data is extracted using [this \(https://dev.twitter.com/overview/api\)](https://dev.twitter.com/overview/api) api. The data contains tweets posted by the following six Twitter accounts: `realDonaldTrump`, `mike_pence`, `GOP`, `HillaryClinton`, `timkaine`, `TheDemocrats`

For every tweet, there are two pieces of information:

- `screen_name` : the Twitter handle of the user tweeting and
- `text` : the content of the tweet.

The tweets have been divided into two parts - train and test available to you in CSV files. For train, both the `screen_name` and `text` attributes were provided but for test, `screen_name` is hidden.

The overarching goal of the problem is to "predict" the political inclination (Republican/Democratic) of the Twitter user from one of his/her tweets. The ground truth (i.e., true class labels) is determined from the `screen_name` of the tweet as follows

- `realDonaldTrump`, `mike_pence`, `GOP` are Republicans
- `HillaryClinton`, `timkaine`, `TheDemocrats` are Democrats

Thus, this is a binary classification problem.

The code to create features from data and to create labels is provided below, you do not need to re-write it. Run the following cells to generate data in a [format \(<https://cloud.google.com/ai-platform/training/docs/algorithms/preprocessing-data>\)](https://cloud.google.com/ai-platform/training/docs/algorithms/preprocessing-data) that the AI Platform can understand.

⚠ DO NOT change this code or use code from HW3

```
In [16]: def create_labels(processed_tweets):  
    return (~processed_tweets['screen_name'].isin(['realDonaldTrump', 'mike_pence',  
  
In [17]: def setup_traintest():  
    # Setup training and testing paths  
    train_path = os.path.join('data', 'tweets_train.csv')  
    train_gcinpath = os.path.join('data', 'train_gcinpath.csv')  
  
    test_path = os.path.join('data', 'tweets_test.csv')  
    test_gcinpath = os.path.join('data', 'test_gcinpath.json')  
  
    # Setup vectorizer and PCA  
    eng_stopwords = set(stopwords.words('english'))  
    vec = TfidfVectorizer(stop_words=eng_stopwords, max_df=0.8, min_df=3, max_features=300)  
    pca = PCA(n_components=300)  
  
    # Setup training data  
    train_df = pd.read_csv(train_path)  
    train_feat = vec.fit_transform(train_df['text'])  
    train_feat = pca.fit_transform(train_feat.todense())  
    train_lab = create_labels(train_df)  
    df_feat = pd.DataFrame(train_feat)  
    df_lab = pd.DataFrame(train_lab)  
    df = pd.concat([df_lab, df_feat], axis=1)  
    df.to_csv(train_gcinpath, index=False, header=False)  
    # Setup testing data  
    test_df = pd.read_csv(test_path)  
    test_feat = vec.transform(test_df['text']).todense()  
    test_feat = pca.transform(test_feat)  
    df = pd.DataFrame(test_feat)  
    def online_prediction_format(row):  
        key = row.name  
        value = str(row.values.tolist()).replace("[", "").replace("]", "")  
        instance = {'key':str(key), 'csv_row':value}  
        return instance  
    df = df.apply(lambda x: online_prediction_format(x), axis=1)  
    df.to_json(test_gcinpath, orient='records', lines=True)
```

```
In [18]: setup_traintest()
```

```
/opt/anaconda3/envs/cs418env/lib/python3.9/site-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeE
rror in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html (https://nump
y.org/doc/stable/reference/generated/numpy.matrix.html)
    warnings.warn(
/opt/anaconda3/envs/cs418env/lib/python3.9/site-packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is deprecated in 1.0 and will raise a TypeE
rror in 1.2. Please convert to a numpy array with np.asarray. For more information
see: https://numpy.org/doc/stable/reference/generated/numpy.matrix.html (https://nump
y.org/doc/stable/reference/generated/numpy.matrix.html)
    warnings.warn(
```

This code should produce two files `train_gcinput.csv` and `test_gcinput.json` in the data subdirectory. `train_gcinput.csv` contains labels in first column and features in remaining columns for each record and `test_gcinput.json` just contains the features and unique key for each record as a JSON instance required by the [prediction](https://cloud.google.com/ai-platform/prediction/docs/overview?hl=en_US) (https://cloud.google.com/ai-platform/prediction/docs/overview?hl=en_US) service. There are 300 features for each record.

Side Note: PCA is a [dimensionality reduction algorithm](https://medium.com/@mayur87545/dimensionality-reduction-1663f960293f) (<https://medium.com/@mayur87545/dimensionality-reduction-1663f960293f>) which makes our file size smaller and our computations simpler.

Uploading Data to Cloud

Before we can train our algorithm, we need to upload `train_gcinput.csv` to the cloud so that the ML engine can access it. But how do we do that?

In *Part 0*, we created a bucket which makes it easier to store files on the cloud and enables different Google Cloud services to access those files. You can think of buckets as folders and all the services on cloud as applications which have access to these folders.

Go to [Google Cloud Console](https://console.cloud.google.com/) (<https://console.cloud.google.com/>) and you should be able to see *Storage* under *Resources* card on project *Dashboard*. Click on it and it will take you to *Storage Browser*. Select your bucket and upload `train_gcinput.csv` in the root of the bucket. Create a new subdirectory in the root of the bucket and name it as `output`, we will use it later to store logs and trained model. It should look like the following figure:

The screenshot shows the Google Cloud Platform Cloud Storage interface. A bucket named "edu-uic-cs418-your_nick_name-homework4" is selected. The bucket details pane shows the location as "us-central1 (Iowa)", storage class as "Standard", public access as "Subject to object ACLs", and protection as "None". The "OBJECTS" tab is active, displaying a list of objects. The list includes a folder named "output/" and a file named "train_gcinput.csv" which is 105.3 MB in size and has a type of "text/csv". There are also buttons for "UPLOAD FILES", "UPLOAD FOLDER", and "CREATE FOLDER" highlighted with blue circles.

Training the Model

Now, we have all the files necessary to train our model. Go to [Google Cloud Console](https://console.cloud.google.com) (<https://console.cloud.google.com>) and from the sidebar, select *AI Platform > Jobs* as shown in the following figure.

The screenshot shows the Google Cloud Platform sidebar. The "AI Platform" menu item is selected and expanded, showing sub-options like "Dashboard", "AI Hub", "Data Labeling", "Pipelines", "Notebooks", and "Jobs". The "Jobs" option is highlighted with a green box. To the right, the "Bucket details" for "edu-uic-cs418-topsykrets-homework4" are displayed, showing the "train_gcinput.csv" file.

If you have not used *Jobs* before then you first need to Enable API to use this service. Click *ENABLE API* (it might take a while).

Machine Learning Engine

Jobs

Google Cloud Machine Learning Engine combines the power of Google's infrastructure with the latest innovations in deep learning.

Use ML Engine Models to create predictive models of your data. To get started, create a model and train different versions of it from the command line.

[Learn more](#)

[ENABLE API](#)

Google cloud provides some built-in models which can be trained on any dataset. Once, the API is enabled, click on *NEW TRAINING JOB* and select *Built-in algorithm training* from the dropdown as shown below:

The screenshot shows the Google Cloud Platform interface for AI Platform. In the top navigation bar, there is a search bar and a dropdown menu set to 'homework4'. Below the navigation bar, there are three main sections: 'AI Platform' (selected), 'Dashboard', and 'Data Labeling'. Under 'AI Platform', there is a 'Jobs' section with a 'NEW TRAINING JOB' button highlighted by a red arrow. A dropdown menu is open from this button, showing two options: 'Built-in algorithm training' and 'Custom code training'. There are also 'REFRESH' and 'CANCEL' buttons in the top right of the dropdown.

We will use a built-in linear classifier for training. Select *Linear Learner* as your training algorithm and click *NEXT*

1 Training algorithm — 2 Training data — 3 Algorithm arguments — 4 Job settings

Before you get started, make sure your training data is in header-less CSV file, prediction target is the first column and the file is stored in a Cloud Storage bucket. [Learn more about how to prepare your data](#)

Select an algorithm *

Linear Learner



[NEXT](#)

[CANCEL](#)

Click **BROWSE** and select `train_gcinput.csv` from the bucket under *Training data path* and setup the remaining arguments as shown in the following screenshot. The validation data is used while training the model for hyper-parameter tuning and we will also use 10% of the training data as Test data for model evaluation. Note that this is not the final test data that we will be using for predictions. Click **NEXT**.

1 Training algorithm — 2 Training data — 3 Algorithm arguments — 4 Job settings

Enable automatic data preprocessing [?](#)

ⓘ Make sure that your data is in a header-less CSV file and the prediction target is in the first column. [Learn more](#)

Training data path *
 gs:// model-bucket-alpha/train_gcinput.csv [BROWSE](#)

The Cloud Storage path where the training data is stored

Validation data

Use a percentage of training data ▾

10 %

Test data (optional)

Use a percentage of training data ▾

10 %

Output directory *
 gs:// model-bucket-alpha/output/ [BROWSE](#)

The path to the Google Cloud Storage location where you want the trained model and other training job output to be stored.

[NEXT](#) [CANCEL](#)

Select model type as classification and leave everything as is, click **NEXT**.

Model Type

classification

Linear Regression or Classification type

Max Steps ?

HyperTune

Learning Rate *

HyperTune

0.001

A scalar used to determine gradient step in gradient descent training. [Learn more](#)

Advanced Section

NEXT

CANCEL

In the job setup, you can specify a *Job ID* and the [type of machine](#) (<https://cloud.google.com/ai-platform/training/docs/machine-types>) you want to run this on. We will run this job on a standard_gpu machine. Note that there are other high-end machines available as well. Click *DONE*.

 High-end machines may incur high computational cost and built-in models do not have the capacity to fully utilize them. [Details](#) (<https://cloud.google.com/ai-platform/training/docs/algorithms/linear-learner>).

Job ID

linear_classifier

Must start with a letter and contain only letters, numbers and underscores. Case-sensitive.

Can't be changed later. 17/128

Region *

us-central1

The Google Cloud Platform region where the training job runs. For efficiency, the region you select should match the region where your training data is stored in Cloud Storage. [Learn more](#)

Scale tier *

CUSTOM

The resources ML Engine allocates to your training job. [Learn more](#)

Custom cluster specification

Master type

standard_gpu

The type of virtual machine to use for your training job's master worker

DONE

CANCEL

Your job should be up and running now and you should be able to see a list of running jobs:

Jobs

NEW TRAINING JOB

BETA

REFRESH

Filter by prefix...

<input type="checkbox"/>	<input checked="" type="radio"/>	Job ID	Type	Create time	Elapsed time	Logs	Labels
<input type="checkbox"/>	<input checked="" type="radio"/>	linear_classifier	Built-in algorithm training		1 sec	View Logs	

You can open a job by clicking on Job ID. This will take you to the job progress page (shown below) where you can monitor resources being used by the job, job logs, parameters and progress. Click on View Logs to view and stream the logs, if something goes wrong then you should be able to see it in the logs.

linear_classifier

 Preparing (16 sec)
Creation time
Start time
End time
Logs View Logs
Training input ▼ SHOW JSON
Training output ▼ SHOW JSON

When the job is completed you should be able to see the following page. It takes about 12-15 minutes.

← Job Details	DEPLOY MODEL	DOWNLOAD MODEL
 Succeeded (11 min 2 sec)		
Creation time		
Start time		
End time		
Logs	View Logs	
TensorBoard	Open Tensorboard	Next Step 1: Evaluate Model
Consumed ML units	0.28	
Training input	▼ SHOW JSON	
Training output	▼ SHOW JSON	
Model location	gs://edu-uic-cs418-topsykretts-homework4/output/model	
CPU GPU NETWORK		

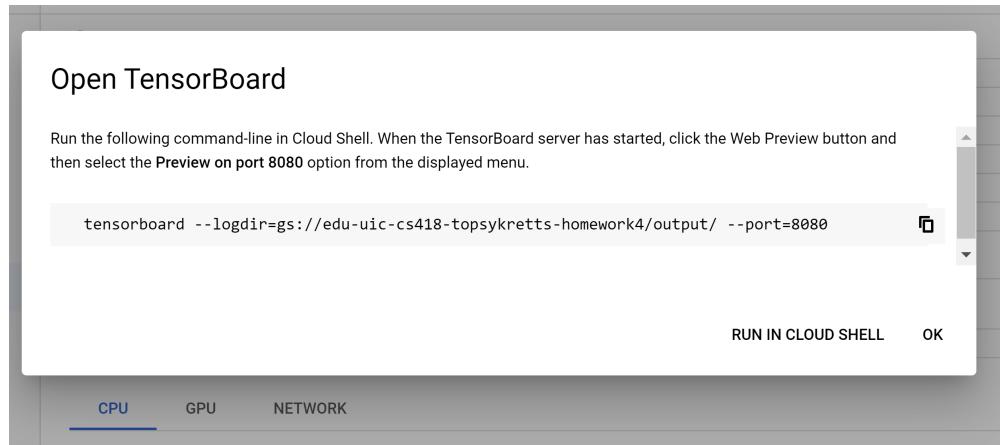
**Next Step 2:
Deploy Model**

**Next Step 1:
Evaluate Model**

Evaluation Metrics and Model Deployment

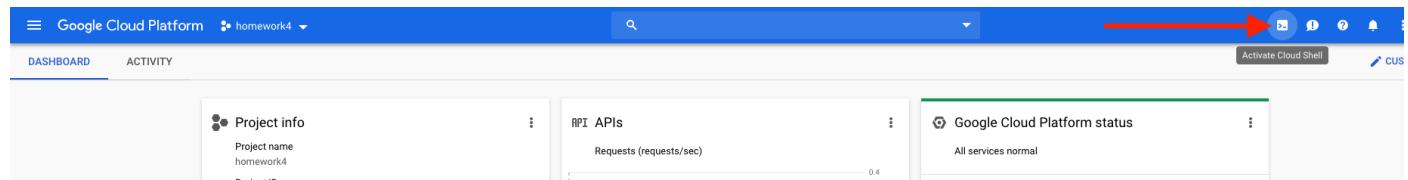
Now that our model is trained, we can see the performance of the model using TensorBoard, a toolkit for ML experimentation. Once we are satisfied with the evaluation, we can deploy the model for online or batch prediction.

Click "Open TensorBoard" link from the completed job details above. You will get a pop-up window.

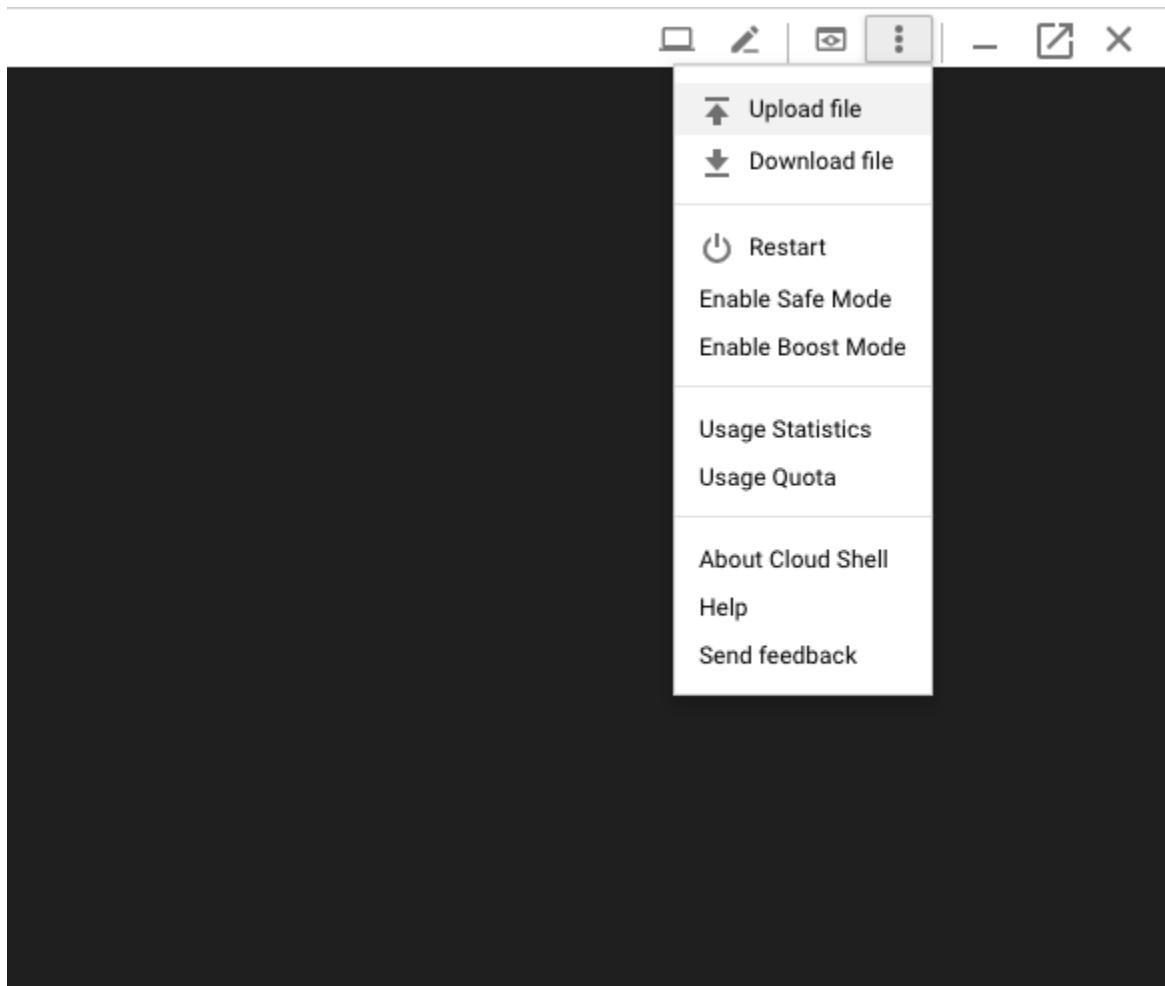


Copy the command and save it to use later. Press "OK".

Next, we setup Google Cloud Shell to access TensorBoard and to make predictions. Click on *Activate Cloud Shell* icon in the top bar as shown in the following screenshot. If you have Cloud Shell already open, click "+" icon to open a new Shell tab.



This should open a terminal in the browser and this is just like a regular linux terminal, you can give a command `ls` to see which files are there. Now, we can upload files to this terminal using the menu on top-right of the terminal window. We need to upload `homework4-key.json` file that we downloaded earlier and `test_gcinput.json` file from `data` directory.



Once the files are uploaded you can verify success using `ls` command.

```
sadhidk9@carbon-virtue-272901:~ (carbon-virtue-272901)$ ls
homework4-key.json README-cloudshell.txt test_gcinput.json
sadhidk9@carbon-virtue-272901:~ (carbon-virtue-272901)$
```

A screenshot of a terminal window titled '(carbon-virtue-272901)'. The window shows the command 'ls' being run, which lists three files: 'homework4-key.json', 'README-cloudshell.txt', and 'test_gcinput.json'. The window has standard OS X-style window controls at the top.

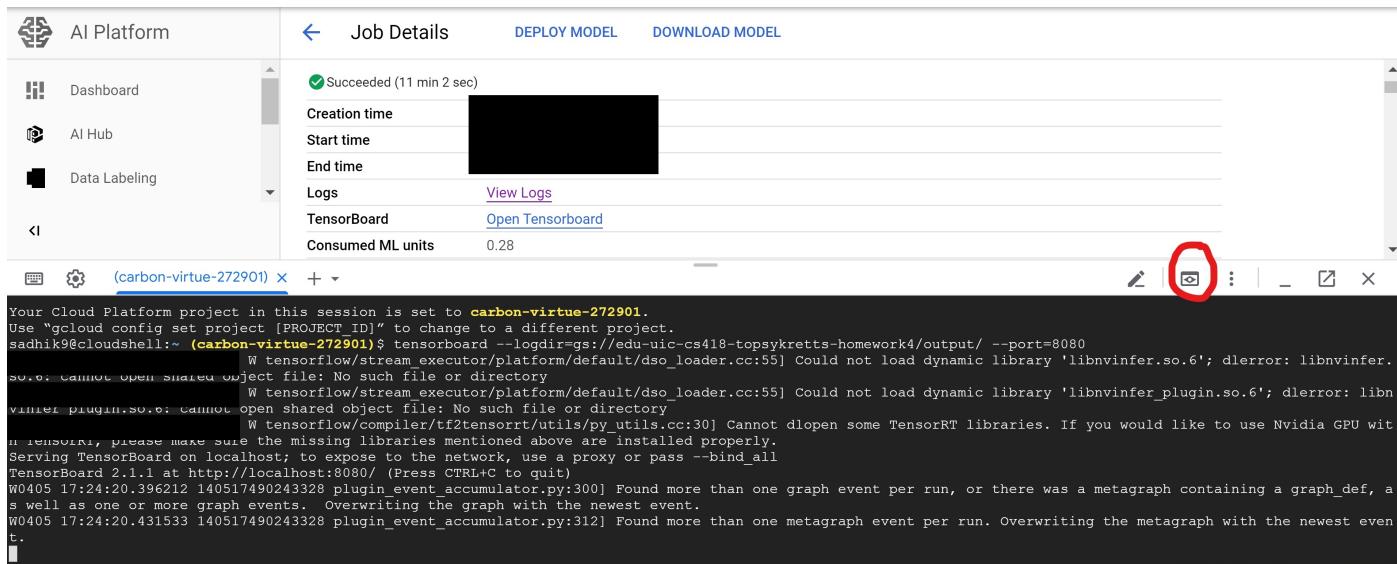
Now, you can run the following command to enable authentication using the key

```
export GOOGLE_APPLICATION_CREDENTIALS=homework4-key.json
```

Now enter the command to start tensorboard you copied earlier.

```
tensorboard --logdir=gs://edu-uic-cs418-your_nick_name-homework4/output/ --port=8080
```

Press Enter and click the "Web Preview" button circled in image below.



Succeeded (11 min 2 sec)

Creation time: [REDACTED]

Start time: [REDACTED]

End time: [REDACTED]

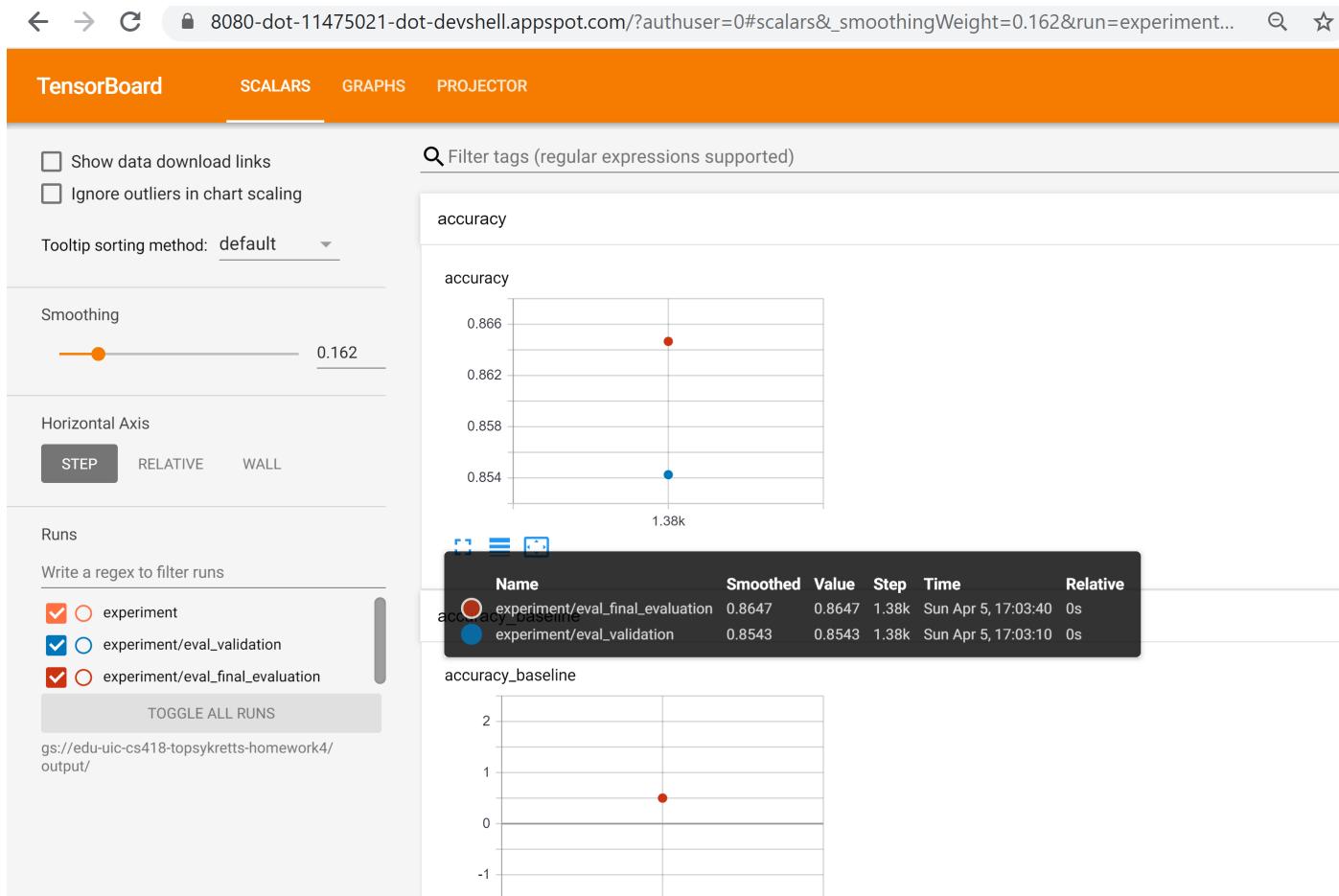
Logs: [View Logs](#)

TensorBoard: [Open Tensorboard](#)

Consumed ML units: 0.28

Your Cloud Platform project in this session is set to **carbon-virtue-272901**.
 Use "gcloud config set project [PROJECT_ID]" to change to a different project.
 sadhik9@cloudshell:~ (carbon-virtue-272901)\$ tensorboard --logdir=gs://edu-uic-cs418-topsykretts-homework4/output/ --port=8080
 W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'libnvinfer.so.6'; dlsym: libnvinfer.
 so.6: cannot open shared object file: No such file or directory
 W tensorflow/stream_executor/platform/default/dso_loader.cc:55] Could not load dynamic library 'libnvinfer_plugin.so.6'; dlsym: libn
 vinfer_plugin.so.6: cannot open shared object file: No such file or directory
 W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:30] Cannot dlopen some TensorRT libraries. If you would like to use Nvidia GPU wit
 h TensorRT, please make sure the missing libraries mentioned above are installed properly.
 Serving TensorBoard on localhost; to expose to the network, use a proxy or pass --bind_all
 TensorBoard 2.1.1 at http://localhost:8080/ (Press CTRL+C to quit)
 W0405 17:24:20.396212 140517490243328 plugin_event_accumulator.py:300] Found more than one graph event per run, or there was a metagraph containing a graph_def, a
 s well as one or more graph events. Overwriting the graph with the newest event.
 W0405 17:24:20.431533 140517490243328 plugin_event_accumulator.py:312] Found more than one metagraph event per run. Overwriting the metagraph with the newest even
 t.

This will launch the TensorBoard interface where you can see the performance metrics of the model and the baseline (majority label classifier).



Now, we are satisfied with our performance and we can deploy it and make predictions using this model on the cloud. Click **DEPLOY MODEL** button from the Job Details. Deploy as a new model and specify a name, click **CONFIRM**.

⚠️ Make sure you uncheck "regional endpoint." It is checked by default.

Deploy Model

Model Name *

linear_classifier_model

Name is permanent, is case-sensitive, must start with a letter, and must only contain letters, numbers and underscores. Model names must be unique within each project.
23 / 128

Use regional endpoint



A regional endpoint provides additional protection for your model against outages in other regions. When you use a regional endpoint, you must send API requests to {region}-ml.googleapis.com instead of ml.googleapis.com. Regional endpoints are currently only available in [certain regions](#) and only work for Compute Engine (N1) machine types (GPUs are allowed). If you need to use legacy (MLS1) machine types, or if you need to deploy to a region without a regional endpoint, unselect this option. Learn more about the [benefits and limitations of regional endpoints](#).

Region *

us-central1

Service availability may vary based on region and model framework. See available regions and services for [TensorFlow](#) and [XGBoost](#)

Description

Enable logging for this model

Enable console logging for this model



Logging settings are permanent for this model. To change your logging

CANCEL

CONFIRM

Specify a version (v1), leave other default (similar to figure below), and click SAVE.

Model to deploy

The exported model of the following training job will be deployed.

Job ID	linear_classifier
Model name	linear_classifier_model

Version name

v1

Name cannot be changed, is case sensitive, must start with a letter and may only contain letters, numbers and underscores. 2/128

Description

Machine type

Single-core CPU

Online prediction deployment

Scaling

Auto-scaling

Minimum number of nodes (optional) 1

Keeping a minimum number of nodes running all the time will avoid dropping requests due to nodes initialisation after the service has scaled down. This setting can increase cost, as you pay for the nodes even when no predictions are served.

SAVE

CLEAR

CANCEL

It will take you to your model page, your model is now being deployed, wait for the green check to appear next to version before you proceed.

Name
linear_classifier_model

Versions

Filter by prefix...

<input type="checkbox"/>	<input checked="" type="radio"/>	Name	Create time	Last used	Labels
<input type="checkbox"/>	<input checked="" type="radio"/>	v1 (default)	[REDACTED]		

Your model is now deployed on the cloud and can be used to make predictions.

Making Predictions

We can either use our local machine to make calls to the Google Cloud and make predictions using gcloud command from [Google Cloud SDK \(<https://cloud.google.com/sdk/docs/>\)](https://cloud.google.com/sdk/docs/) or we can use [Google Cloud Shell \(<https://cloud.google.com/shell/>\)](https://cloud.google.com/shell/) which comes with Google Cloud SDK. We will use *Google Cloud Shell* in this tutorial.

Visit [Google Cloud Console \(<https://console.cloud.google.com>\)](https://console.cloud.google.com) and click on *Activate Cloud Shell* icon in the top bar like you did earlier. If you have Cloud Shell already open, click "+" icon to open a new Shell tab.

You can use `ls` command to see if the uploaded files are there.

Again run the following command to enable authentication using the key

```
export GOOGLE_APPLICATION_CREDENTIALS=homework4-key.json
```

Once, the authentication is enabled, you can use the following `gcloud` command to make predictions:

```
gcloud ai-platform predict --model linear_classifier_model --version v1 --json-instances test_gcinput.json --region global
```

If everything goes right, then you should be able to see the following output. Here, *CLASSES* are just binary class labels 0/1 and *LOGISTIC* shows the probability of being in that class.

```
KEY: ['98']
LOGISTIC: [0.4374176263809204]
LOGITS: [-0.25164908170700073]
PROBABILITIES: [0.5625823736190796, 0.4374176859855652]

ALL_CLASS_IDS: [0, 1]
ALL_CLASSES: ['0', '1']
CLASS_IDS: [1]
CLASSES: ['1']
KEY: ['99']
LOGISTIC: [0.5849515795707703]
LOGITS: [0.343133807182312]
PROBABILITIES: [0.41504842042922974, 0.5849515795707703]
```

Let us save these results in JSON format using this command.

```
gcloud ai-platform predict --model linear_classifier_model --version v1 --js
on-instances test_gcinput.json --format=json --region global > results.json
```

This should create a file `results.json` in current directory. You can view the produced JSON using this command:

```
cat results.json
```

 Select *Download File* from the top-right menu and save `results.json` in your `sub` (submission) subdirectory and that's it!

Before Submitting

 Make sure that you have all the required screenshots and data files in `sub` subdirectory signified by this icon.

 Make sure that you have made all necessary additions to the notebook signified by this icon.

 Make sure that you have read all the warnings.

Finally,

 Once the homework is fully graded, you may remove your project from Google cloud or stop any services to avoid running expenses. Most likely there will be no running expenses even if you do not stop/remove the project but it is usually not recommended to keep unnecessary components running on Google Cloud as it may incur additional cost.

 Before submitting written part of the HW, make sure that all parts of your solution are completely visible, pay special attention to the images you have included.

Submission

As instructed earlier on the top of this notebook, submit all files in the `sub` directory and `hw4.ipynb` file to `Homework4 - code` assignment in the Gradescope. Similarly, submit the exported PDF `hw4.pdf` to `Homework4 - written` assignment in the Gradescope. Make sure all the images you added are clearly

visible, the links are not broken, code is not truncated, and answers are marked appropriately.