

# Assignment 1 - Probability, Linear Algebra, & Computational Programming

Version 2: Updated 1/14/24 (Question 4 revised)

*Daniela Jimenez Lara*

Netid: dj216

Note: this assignment falls under collaboration Mode 2: Individual Assignment – Collaboration Permitted. Please refer to the syllabus for additional information.

Instructions for all assignments can be found [here](#), and are also linked to from the course syllabus.

Total points in the assignment add up to 90; an additional 10 points are allocated to professionalism and presentation quality.

## Learning Objectives

The purpose of this assignment is to provide a refresher on fundamental concepts that we will use throughout this course and provide an opportunity to develop skills in any of the related skills that may be unfamiliar to you. Through the course of completing this assignment, you will...

- Refresh your knowledge of probability theory including properties of random variables, probability density functions, cumulative distribution functions, and key statistics such as mean and variance.
- Revisit common linear algebra and matrix operations and concepts such as matrix multiplication, inner and outer products, inverses, the Hadamard (element-wise) product, eigenvalues and eigenvectors, orthogonality, and symmetry.
- Practice numerical programming, core to machine learning, by loading and filtering data, plotting data, vectorizing operations, profiling code speed, and debugging and optimizing performance. You will also practice computing probabilities based on simulation.
- Develop or refresh your knowledge of Git version control, which will be a core tool used in the final project of this course
- Apply your skills altogether through an exploratory data analysis to practice data cleaning, data manipulation, interpretation, and communication

We will build on these concepts throughout the course, so use this assignment as a catalyst to deepen your knowledge and seek help with anything unfamiliar.

If some references would be helpful on these topics, I would recommend the following resources:

- [Mathematics for Machine Learning](#) by Deisenroth, Faisal, and Ong
- [Deep Learning](#); Part I: Applied Math and Machine Learning Basics by Goodfellow, Bengio, and Courville
- [The Matrix Calculus You Need For Deep Learning](#) by Parr and Howard
- [Dive Into Deep Learning](#); Appendix: Mathematics for Deep Learning by Weness, Hu, et al.

*Note: don't worry if you don't understand everything in the references above - some of these books dive into significant minutia of each of these topics.*

---

## Probability and Statistics Theory

*Note: for all assignments, write out equations and math using markdown and [LaTeX](#). I recommend that you complete the work on paper before typing up the final version. For this assignment show your math for questions 1-4, meaning that you should include any intermediate steps necessary to understand the logic of your solution. Most can be completed in 3-4 steps. Being proficient in expressing yourself clearly, sometimes mathematically, is a valuable skill to have as a data scientist*

1

**[3 points]**

Let  $f(x) = \begin{cases} 0 & x < 0 \\ \alpha x^2 & 0 \leq x \leq 2 \\ 0 & 2 < x \end{cases}$  For what value of  $\alpha$  is  $f(x)$  a valid probability density function?

$$f(x) \begin{cases} 0 & x < 0 \\ \alpha x^2 & 0 \leq x \leq 2 \\ 0 & 2 < x \end{cases}$$

$$\int_0^2 \alpha x^2 = 1$$

**ANSWER**

$$\alpha \frac{x^3}{3} = 1$$

$$\alpha \frac{2^3}{3} - \frac{2^2}{3}$$

$$\alpha \cdot \frac{8}{3} = 1 \frac{3}{8}$$

$$\alpha = \frac{3}{8}$$

**2**

**[3 points]** What is the cumulative distribution function (CDF) that corresponds to the following probability distribution function? Please state the value of the CDF for all possible values of  $x$ .

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

**ANSWER**

$$\int_0^x \frac{1}{3} dx = \frac{x}{3}$$

$$x \left[ \frac{x}{3} \right] =$$

$$= \frac{x}{3} - \frac{0}{3} = \frac{x}{3}$$

$$f(x) = \begin{cases} 0 & x < 0 \\ \frac{x}{3} & 0 < x < 3 \\ 1 & x > 3 \end{cases}$$

**3**

**[6 points]** For the probability distribution function for the random variable  $X$ ,

$$f(x) = \begin{cases} \frac{1}{3} & 0 < x < 3 \\ 0 & \text{otherwise} \end{cases}$$

what is the (a) expected value and (b) variance of  $X$ . *Show all work.*

### ANSWER

Expected value of  $x$ :

$$E[x] = \int_0^3 x \cdot \frac{1}{3} dx$$

$$\frac{1}{3} \left[ \frac{x^2}{2} \right]_0^3$$

$$\frac{1}{3} \left( \frac{9}{2} \right) = \frac{9}{6} = \frac{3}{2}$$

$$E[X] = \frac{3}{2}$$

Variance of  $x$ :

$$\sigma = E[(x - \mu)^2] = \int_{-\infty}^{\infty} (x - \mu)^2 f(x) dx, \text{ where:}$$

$$E[(x - \mu)^2] = E(x^2) - [E(x)]^2$$

$$= \int_{-\infty}^{\infty} x^2 f(x) dx - [E(x)]^2$$

$$= \int_0^3 x^2 \frac{1}{3} dx - \left( \frac{3}{2} \right)^2$$

$$= \frac{1}{3} \left[ \frac{x^3}{3} \right]_0^3 - \frac{9}{4} =$$

$$= \frac{1}{3} \left( \frac{27}{3} \right) - \frac{9}{4} =$$

$$= \frac{9}{3} - \frac{9}{4} = \frac{36}{12} - \frac{27}{12} = \frac{9}{12} = \frac{3}{4}$$

$$\text{Var}[X] = \frac{3}{4}$$

## 4

**[6 points]** You are given the training data below and asked to determine the probability that a sample of  $x = 0.54$  comes from class 1, or equivalently,  $P(Y = 1 | X = 0.54)$ . The feature,  $x$ , can take on real values between 0 and 1.

$x$ value range	Negative data samples ( $x, y = 0$ )	Positive data samples ( $x, y = 1$ )
0.0 - 0.1	(0.05,0),(0.07,0)	None
0.1 - 0.2	(0.11,0),(0.13,0),(0.19,0)	(0.14,1)

<i>x</i> value range	Negative data samples ( <i>x</i> , <i>y</i> = 0)	Positive data samples ( <i>x</i> , <i>y</i> = 1)
0.2 - 0.3	(0.23,0)	(0.24,1)
0.3 - 0.4	(0.35,0), (0.37,0)	(0.32,1)
0.4 - 0.5	(0.49,0)	(0.47,1)
0.5 - 0.6	(0.51,0)	(0.53,1)
0.6 - 0.7	None	(0.61,1)
0.7 - 0.8	None	(0.77,1)
0.8 - 0.9	None	(0.83,1)
0.9 - 1.0	None	(0.92,1), (0.98,1)

Note: You don't need to use these data directly, but this provides an example of how the data could be distributed to form the empirical likelihoods and priors shown below.

You're likely familiar with Bayes' Rule, which for discrete random variables states that:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Additionally,  $P(X) = \sum_y P(X|Y = y)P(Y = y)$


In our case, however, the variable *Y* is discrete but the variable *X* is continuous, so we express this function a bit differently. We can compute the poster probability,  $P(Y|X)$ , based on the likelihood function of the data conditioned on the class of the samples,  $f_{X|Y}(x, y)$ , the prior  $P(Y)$  which is essentially the distribution of the class labels across all the data, and the evidence  $f_X(x)$  which is the probability distribution function of the features, *X*, regardless of class labels:


$$P(Y = y|X = x) = \frac{f_{X|Y}(x, y)P(Y = y)}{f_X(x)}$$

Also, note that  $f_X(x)$  can be computed in this case (due to the discrete *Y* values of 0 and 1) as:

$$f_X(x) = f_{X|Y}(x, y = 0)P(Y = 0) + f_{X|Y}(x, y = 1)P(Y = 1)$$

Below are the prior and the likelihood functions based on the dataset above. Note that here we use  $P(\cdot)$  to note a probability and  $f(\cdot)$  to note a probability distribution function.

 No description has been provided for this image

 No description has been provided for this image

1. What is  $f_{X|Y}(x = 0.54, y = 1)P(Y = 1)$ ?
2. What is  $f_X(x = 0.54)$ ?
3. What is  $P(Y = 1|X = 0.54)$ ?

Show each value you use for each computation.

**ANSWER**

---

## Linear Algebra

### 5

**[5 points]** A common task in machine learning is a change of basis: transforming the representation of our data from one space to another. A prime example of this is through the process of dimensionality reduction as in Principle Components Analysis where we often seek to transform our data from one space (of dimension  $n$ ) to a new space (of dimension  $m$ ) where  $m < n$ . Assume we have a sample of data of dimension  $n = 4$  (as shown below) and we want to transform it into a dimension of  $m = 2$ .

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

- (a) What are the dimensions of a matrix,  $\mathbf{A}$ , that would linearly transform our sample of data,  $\mathbf{x}$ , into a space of  $m = 2$  through the operation  $\mathbf{Ax}$ ?
- (b) Express this transformation in terms of the components of  $\mathbf{x}$ :  $x_1, x_2, x_3, x_4$  and the matrix  $\mathbf{A}$  where each entry in the matrix is denoted as  $a_{i,j}$  (e.g. the entry in the first row and second column would be  $a_{1,2}$ ). Your answer will be in the form of a matrix expressing result of the product  $\mathbf{Ax}$ .

*Note: please write your answers here in LaTeX*

**ANSWER**

- (a) the dimensions of matrix  $\mathbf{A}$  would be 4 by 2

$$(b) \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{Ax} = \begin{bmatrix} x_1 a_{11} & +x_1 a_{12} & +x_1 a_{13} & +x_1 a_{14} \\ x_2 a_{21} & +x_2 a_{22} & +x_2 a_{23} & +x_2 a_{24} \end{bmatrix}$$

## 6

**[14 points] Matrix manipulations and multiplication.** Machine learning involves working with many matrices, so this exercise will provide you with the opportunity to practice those skills.

$$\text{Let } \mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -1 \\ 3 \\ 8 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 4 \\ -3 \\ 6 \end{bmatrix}, \text{ and } \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Compute the following **using Python** or indicate that it cannot be computed. Refer to NumPy's tools for handling matrices. While all answers should be computer using Python, your response to whether each item can be computed should refer to underlying linear algebra. There may be circumstances when Python will produce an output, but based on the dimensions of the matrices involved, the linear algebra operation is not possible. **For the case when an operation is invalid, explain why it is not.**

When the quantity can be computed, please provide both the Python code AND the output of that code (this need not be in LaTeX)

1.  $\mathbf{AA}$
2.  $\mathbf{AA}^T$
3.  $\mathbf{Ab}$
4.  $\mathbf{Ab}^T$
5.  $\mathbf{bA}$
6.  $\mathbf{b}^T \mathbf{A}$
7.  $\mathbf{bb}$
8.  $\mathbf{b}^T \mathbf{b}$
9.  $\mathbf{bb}^T$
10.  $\mathbf{b} + \mathbf{c}^T$
11.  $\mathbf{b}^T \mathbf{b}^T$
12.  $\mathbf{A}^{-1} \mathbf{b}$
13.  $\mathbf{A} \circ \mathbf{A}$
14.  $\mathbf{b} \circ \mathbf{c}$

Note: The element-wise (or Hadamard) product is the product of each element in one matrix with the corresponding element in another matrix, and is represented by the symbol " $\circ$ ".

## ANSWER

```
In [ ]: import numpy as np
        from numpy.linalg import inv

        A = np.array([(1, 2, 3), (2, 4, 5), (3, 5, 6)])
        At = A.T
        Ainv = inv(A)
        B = np.array([[-1], [3], [8]])
        Bt = B.T
        C = np.array([[4], [-3], [6]])
        Ct = C.T
        I = np.array([(1, 0, 0), (0, 1, 0), (0, 0, 1)])
```

- 1.  $\mathbf{A}\mathbf{A}$  is possible to compute because both matrixes are the same and hence the first one has the same number of columns as the second has rows.

```
In [ ]: A @ A
```

```
Out[ ]: array([[14, 25, 31],
               [25, 45, 56],
               [31, 56, 70]])
```

- 2.  $\mathbf{A}\mathbf{A}^T$  is possible to compute because the transposed matrix has the same dimensions as the original matrix, hence, the first one has the same number of columns as the second has rows.

```
In [ ]: A @ At
```

```
Out[ ]: array([[14, 25, 31],
               [25, 45, 56],
               [31, 56, 70]])
```

- 3.  $\mathbf{A}\mathbf{b}$  is possible to compute as the first matrix has the same number of columns as the second has rows.

```
In [ ]: A @ B
```

```
Out[ ]: array([[29],
               [50],
               [60]])
```

- 4.  $\mathbf{A}\mathbf{b}^T$  the operation is invalid as the first matrix has a different number of columns as the second one has rows.



- 5.  $\mathbf{bA}$  the operation is invalid as the first matrix has a different number of columns as the second one has rows
- 6.  $\mathbf{b}^T \mathbf{A}$  the operation is valid as the first matrix has the same number of columns as the second one has rows

In [ ]: `Bt @ A`

Out[ ]: `array([[29, 50, 60]])`

- 7.  $\mathbf{bb}$  the operation is invalid as the first matrix has a different number of columns than the second one has rows
- 8.  $\mathbf{b}^T \mathbf{b}$  the operation is valid as the first matrix has the same number of columns than the second one has rows

In [ ]: `Bt @ B`

Out[ ]: `array([[74]])`

- 9.  $\mathbf{bb}^T$  the operation is valid as the first matrix has the same number of columns than the second one has rows

In [ ]: `B @ Bt`

Out[ ]: `array([[ 1, -3, -8],  
[-3, 9, 24],  
[-8, 24, 64]])`

- 10.  $\mathbf{b} + \mathbf{c}^T$  the operation is valid as the first matrix has the same number of columns than the second one has rows

In [ ]: `B @ Ct`

Out[ ]: `array([[ -4, 3, -6],  
[ 12, -9, 18],  
[ 32, -24, 48]])`

- 11.  $\mathbf{b}^T \mathbf{b}^T$  the operation is invalid as the first matrix has a different number of columns than the second one has rows
- 12.  $\mathbf{A}^{-1} \mathbf{b}$  the operation is valid as the first matrix has the same number of columns than the second one has rows

```
In [ ]: Ainv @ B
```

```
Out[ ]: array([[ 6.],
               [ 4.],
               [-5.]])
```

- 13.  $\mathbf{A} \circ \mathbf{A}$  is possible as both matrixes share the same dimensions

```
In [ ]: A * A
```

```
Out[ ]: array([[ 1,  4,  9],
               [ 4, 16, 25],
               [ 9, 25, 36]])
```

- 14.  $\mathbf{b} \circ \mathbf{c}$  is possible as both matrixes share the same dimensions

```
In [ ]: B * C
```

```
Out[ ]: array([[ -4],
               [ -9],
               [48]])
```

## 7

**[8 points] Eigenvectors and eigenvalues.** Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. They are used extensively in machine learning and in this course we will encounter them in relation to Principal Components Analysis (PCA), clustering algorithms, For an intuitive review of these concepts, explore this [interactive website at Setosa.io](#). Also, the series of linear algebra videos by Grant Sanderson of 3Brown1Blue are excellent and can be viewed on youtube [here](#). For these questions, numpy may once again be helpful.

1. Calculate the eigenvalues and corresponding eigenvectors of matrix  $\mathbf{A}$  above, from the last question.
2. Choose one of the eigenvector/eigenvalue pairs,  $\mathbf{v}$  and  $\lambda$ , and show that  $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$ . This relationship extends to higher orders:  $\mathbf{A}\mathbf{A}\mathbf{v} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for eigenvectors from real, symmetric matrices. In three dimensions or less, this means that the eigenvectors are perpendicular to each other. Typically we use the orthogonal basis of our standard x, y, and z, Cartesian coordinates, which allows us, if we combine them linearly, to represent any point in a 3D space. But any three orthogonal vectors can do the same. We will see this property is used in PCA to identify the dimensions of greatest variation in our data when we discuss dimensionality reduction.

**ANSWER**

a

```
In [ ]: A = np.array([(1, 2, 3), (2, 4, 5), (3, 5, 6)])

eigenvalues, eigenvectors = np.linalg.eig(A)

print(f"The eigenvalues are: {eigenvalues}")
print(f"The eigenvectors are: {eigenvectors}")
```

The eigenvalues are: [11.34481428 -0.51572947 0.17091519]  
 The eigenvectors are: [[-0.32798528 -0.73697623 0.59100905]  
 [-0.59100905 -0.32798528 -0.73697623]  
 [-0.73697623 0.59100905 0.32798528]]

b

```
In [ ]: eigenvector_1 = eigenvectors[:, 1]
eigenvalue_1 = eigenvalues[1]
# A * v
A_v = np.dot(A, eigenvector_1)

# lambda * v
lambda_v = eigenvalue_1 * eigenvector_1

# Print the results
print(f"A * v: {A_v}")
print(f"lambda * v: {lambda_v}")
```

A \* v: [ 0.38008036 0.16915167 -0.30480078]  
 lambda \* v: [ 0.38008036 0.16915167 -0.30480078]

c

```
In [ ]: eiv_0 = eigenvectors[:, 0]
eiv_1 = eigenvectors[:, 1]
eiv_2 = eigenvectors[:, 2]

product_12 = np.dot(eiv_0, eiv_1)
product_13 = np.dot(eiv_0, eiv_2)
product_23 = np.dot(eiv_1, eiv_2)

print(f"Product v1 and v2: {product_12}")
print(f"Product v1 and v3: {product_13}")
print(f"Product v2 and v3: {product_23}")
```

Product v1 and v2: 3.5330797728269967e-16  
 Product v1 and v3: -4.778175388788784e-16  
 Product v2 and v3: -6.1377913704128e-16

Since our matrix  $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$  is symetrical, the inner product of its eigenvectors are close to zero, making it orthogonal

---

## Numerical Programming

### 8

**[10 points]** Loading data and gathering insights from a real dataset

In data science, we often need to have a sense of the idiosyncrasies of the data, how they relate to the questions we are trying to answer, and to use that information to help us to determine what approach, such as machine learning, we may need to apply to achieve our goal. This exercise provides practice in exploring a dataset and answering question that might arise from applications related to the data.

**Data.** The data for this problem can be found in the `data` subfolder in the `assignments` folder on [github](#). The filename is `a1_egrid2016.xlsx`. This dataset is the Environmental Protection Agency's (EPA) [Emissions & Generation Resource Integrated Database \(eGRID\)](#) containing information about all power plants in the United States, the amount of generation they produce, what fuel they use, the location of the plant, and many more quantities. We'll be using a subset of those data.

The fields we'll be using include:

field	description
SEQPLT16	eGRID2016 Plant file sequence number (the index)
PSTATABB	Plant state abbreviation
PNAME	Plant name
LAT	Plant latitude
LON	Plant longitude
PLPRMFL	Plant primary fuel
CAPFAC	Plant capacity factor
NAMEPCAP	Plant nameplate capacity (Megawatts MW)
PLNGENAN	Plant annual net generation (Megawatt-hours MWh)
PLCO2EQA	Plant annual CO2 equivalent emissions (tons)

For more details on the data, you can refer to the [eGrid technical documents](#). For example, you may want to review page 45 and the section "Plant Primary Fuel (PLPRMFL)", which gives the full names of the fuel types including WND for wind, NG for natural gas, BIT for Bituminous coal, etc.

There also are a couple of "gotchas" to watch out for with this dataset:

- The headers are on the second row and you'll want to ignore the first row (they're more detailed descriptions of the headers).
- NaN values represent blanks in the data. These will appear regularly in real-world data, so getting experience working with these sorts of missing values will be important.

**Your objective.** For this dataset, your goal is to answer the following questions about electricity generation in the United States:

**(a)** Which plant has generated the most energy (measured in MWh)?

**(b)** What is the name of the northern-most power plant in the United States?

**(c)** What is the state where the northern-most power plant in the United States is located?

**(d)** Plot a bar plot showing the amount of energy produced by each fuel type across all plants.

**(e)** From the plot in (d), which fuel for generation produces the most energy (MWh) in the United States?

## ANSWER

```
In [ ]: import pandas as pd

bd = pd.read_excel("a1_egrid2016.xlsx", skiprows=1)
```

- **(a)** The plant that has generated the most energy measured in Kwh is Palo Alto

```
In [ ]: bd[["PNAME", "PLNGENAN"]].sort_values("PLNGENAN", ascending=False)
```

Out [ ]:

	PNAME	PLNGENAN
<b>390</b>	Palo Verde	3.237748e+07
<b>164</b>	Browns Ferry	2.621462e+07
<b>7966</b>	Peach Bottom	2.187544e+07
<b>8764</b>	South Texas Project	2.169430e+07
<b>8157</b>	Oconee	2.117710e+07
...	...	...
<b>9683</b>	Osage (WY)	NaN
<b>9684</b>	Pilot Butte	NaN
<b>9694</b>	Simpson Ridge Wind Farm LLC	NaN
<b>9698</b>	Sweetwater Solar	NaN
<b>9703</b>	Two Elk Generating Station	NaN

9709 rows × 2 columns

- **(b)** The name of the northern-most power plant in the United States is Barrow
- **(c)** The state where the northern-most power plant in the United States is Arkansas

In [ ]: `bd[["PNAME", "PSTATABB", "LAT", "LON"]].sort_values("LAT", ascending=False)`

Out [ ]:

	PNAME	PSTATABB	LAT	LON
<b>11</b>	Barrow	AK	71.292000	-156.778600
<b>93</b>	NSB Wainwright Utility	AK	70.642877	-160.020461
<b>88</b>	NSB Atkasuk Utility	AK	70.482600	-157.425200
<b>131</b>	TNSG North Plant	AK	70.235278	-148.383611
<b>90</b>	NSB Nuiqsut Utility	AK	70.220565	-150.993492
...	...	...	...	...
<b>8809</b>	Turbine	TX	NaN	NaN
<b>9349</b>	SPI - Everett	WA	NaN	NaN
<b>9574</b>	Wheaton Solar	WI	NaN	NaN
<b>9596</b>	Chemours Belle Plant	WV	NaN	NaN
<b>9631</b>	UCC South Charleston Plant	WV	NaN	NaN

9709 rows × 4 columns

**(d)** Plot a bar plot showing the amount of energy produced by each fuel type across all plants.

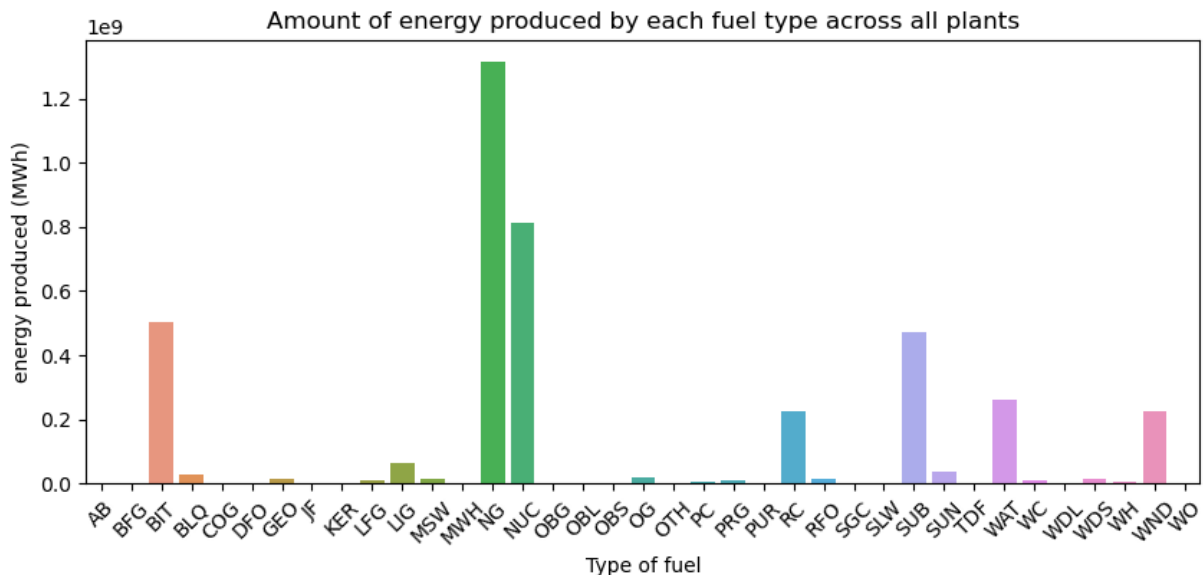
```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter(action="ignore", category=FutureWarning)

bd2 = bd.groupby("PLPRMFL")["PLNGENAN"].sum().reset_index()

# sns.set_theme(style="whitegrid")
plt.figure(figsize=(10, 4))
sns.barplot(x="PLPRMFL", y="PLNGENAN", data=bd2)
plt.xticks(rotation=45)
plt.xlabel("Type of fuel")
plt.ylabel("energy produced (MWh)")
plt.title("Amount of energy produced by each fuel type across all plants")
```

Out[ ]: Text(0.5, 1.0, 'Amount of energy produced by each fuel type across all plants')



**(e)** Natural gas fuel (NG) produces the most energy in the United states

## 9

**[6 points]** *Vectorization*. When we first learn to code and think about iterating over an array, we often use loops. If implemented correctly, that does the trick. In machine learning, we iterate over so much data that those loops can lead to significant slow downs if they are not computationally efficient. In Python, vectorizing code and relying on matrix operations with efficient tools like numpy is typically the faster approach. Of

course, numpy relies on loops to complete the computation, but this is at a lower level of programming (typically in C), and therefore is much more efficient. This exercise will explore the benefits of vectorization. Since many machine learning techniques rely on matrix operations, it's helpful to begin thinking about implementing algorithms using vector forms.

Begin by creating an array of 10 million random numbers using the numpy `random.randn` module. Compute the sum of the squares of those random numbers first in a for loop, then using Numpy's `dot` module to perform an inner (dot) product. Verify that your code produces the same output in each case. Time how long it takes to compute each and report the results and report the output. How many times faster is the vectorized code than the for loop approach? (Note - your results may vary from run to run).

Your output should use the `print()` function as follows (where the # symbols represent your answers, to a reasonable precision of 4-5 significant figures):

```
Time [sec] (non-vectorized): #####
```

```
Time [sec] (vectorized): #####
```

```
The vectorized code is ##### times faster than the nonvectorized code
```

## ANSWER

```
In [ ]: import time

# sum of squares using for loop (non vectorized)
# create 10 mill array
mil_array = np.random.randn(10000000)
# loop
sum_sq_loop = 0
start_loop = time.time()
for i in mil_array:
    i_sq = i**2
    sum_sq_loop += i_sq
end_loop = time.time()
time_loop = round(end_loop - start_loop, 5)

# print(f'sum loop:{sum_sq_loop},time: {time_loop}')
```

```
# vector
start_vector = time.time()
sum_sq_vector = np.sum(mil_array * mil_array)
end_vector = time.time()
time_vector = round(end_vector - start_vector, 5)

# dif in times
times_fast = round(time_loop / time_vector, 2)
```



```
# print(f'sum vector:{sum_sq_vector},time: {time_vector}')

print(
    f"Time [sec] (non-vectorized): {time_loop}\nTime [sec] (vectorized):{time_vector}"
)
```

Time [sec] (non-vectorized): 0.1523

Time [sec] (vectorized):0.00099

The vectorized code is 153.84 times faster than the non vectorized code

## 10

**[10 points]** This exercise will walk through some basic numerical programming and probabilistic thinking exercises, two skills which are frequently used in machine learning for answering questions from our data.

1. Synthesize  $n = 10^4$  normally distributed data points with mean  $\mu = 2$  and a standard deviation of  $\sigma = 1$ . Call these observations from a random variable  $X$ , and call the vector of observations that you generate,  $\mathbf{x}$ .
2. Calculate the mean and standard deviation of  $\mathbf{x}$  to validate (1) and provide the result to a precision of four significant figures.
3. Plot a histogram of the data in  $\mathbf{x}$  with 30 bins
4. What is the 90th percentile of  $\mathbf{x}$ ? The 90th percentile is the value below which 90% of observations can be found.
5. What is the 99th percentile of  $\mathbf{x}$ ?
6. Now synthesize  $n = 10^4$  normally distributed data points with mean  $\mu = 0$  and a standard deviation of  $\sigma = 3$ . Call these observations from a random variable  $Y$ , and call the vector of observations that you generate,  $\mathbf{y}$ .
7. Create a new figure and plot the histogram of the data in  $\mathbf{y}$  on the same axes with the histogram of  $\mathbf{x}$ , so that both histograms can be seen and compared.
8. Using the observations from  $\mathbf{x}$  and  $\mathbf{y}$ , estimate  $E[XY]$

### ANSWER

1 and 2

```
In [ ]: np.random.seed(33)
n = 10**4
mu = 2
sigma = 1
x = np.random.normal(mu, sigma, n)

mean_x = np.mean(x)
std_x = np.std(x)
```

```
print(f"Mean x: {mean_x:.4f}")
print(f"Standard Deviation x: {std_x:.4f}")
```

Mean of x: 2.0026

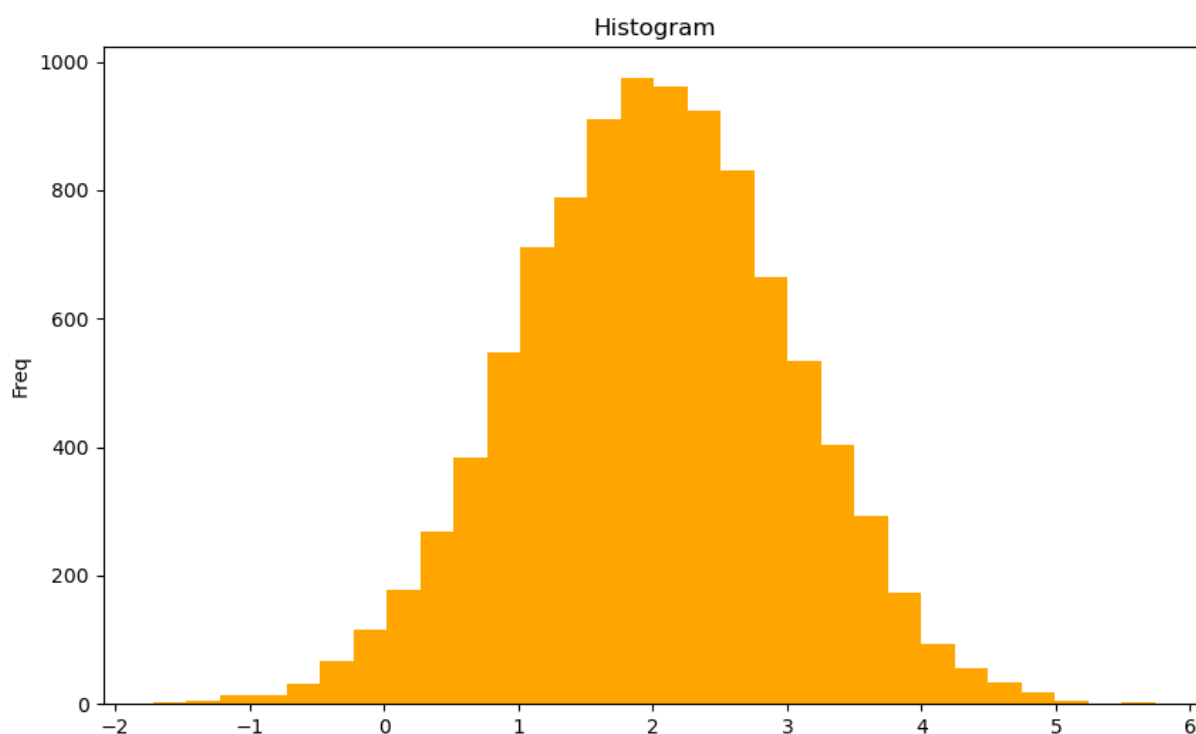
Standard Deviation of x: 1.0025

3

```
In [ ]: plt.figure(figsize=(10, 6))
plt.hist(x, bins=30, color="orange")

plt.ylabel("Freq")
plt.title("Histogram")

# Show the plot
plt.show()
```



4 and 5

```
In [ ]: percentile_90 = np.percentile(x, 90)
percentile_99 = np.percentile(x, 99)
print(
    f"The 90th percentile, where the value below which 90% of observations c
)
print(
    f"The 99th percentile, where the value below which 99% of observations c
)
```

The 90th percentile, where the value below which 90% of observations can be found, is: 3.2908

The 99th percentile, where the value below which 99% of observations can be found, is: 4.3249

6 and 7

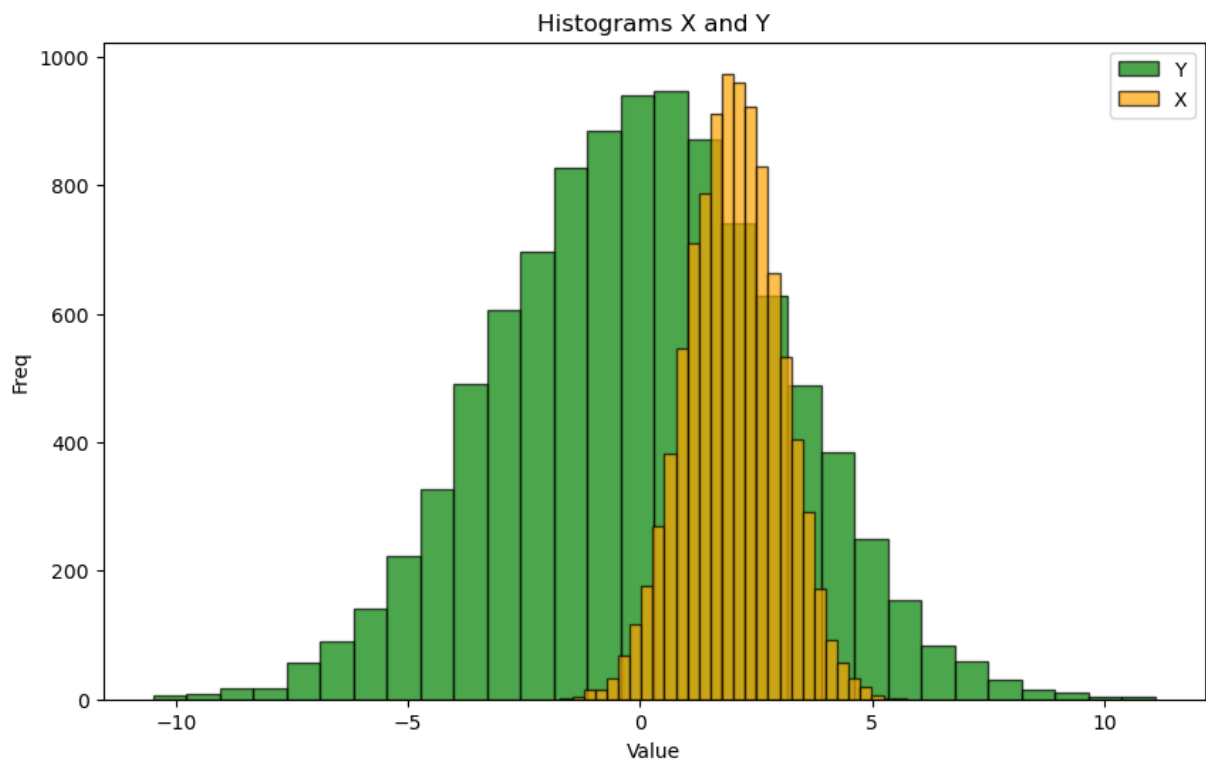
```
In [ ]: n_2 = 10**4
mu_2 = 0
sigma_2 = 3

y = np.random.normal(mu_2, sigma_2, n)

plt.figure(figsize=(10, 6))
plt.hist(y, bins=30, color="green", alpha=0.7, label="Y", edgecolor="black")
plt.hist(x, bins=30, color="orange", alpha=0.7, label="X", edgecolor="black")

# Set labels and title
plt.xlabel("Value")
plt.ylabel("Freq")
plt.title("Histograms X and Y")
plt.legend()

# Show the plot
plt.show()
```



8. estimate  $E[XY]$

```
In [ ]: e_xy = np.mean(x * y)

print(f"Estimated E[XY]: {e_xy:.4f}")
```

Estimated E[XY]: 0.0109

---

## Version Control via Git

### 11

**[4 points]** Git is efficient for collaboration, and expectation in industry, and one of the best ways to share results in academia. You can even use some Git repositories (e.g. Github) as hosts for website, such as with the [course website](#). As a data scientist with experience in machine learning, Git is expected. We will interact with Git repositories (a.k.a. repos) throughout this course, and your project will require the use of git repos for collaboration.

Complete the [Atlassian Git tutorial](#), specifically the following listed sections. Try each concept that's presented. For this tutorial, instead of using BitBucket as your remote repository host, you may use your preferred platform such as [Github](#) or [Duke's Gitlab](#).

1. [What is version control](#)
2. [What is Git](#)
3. [Install Git](#)
4. [Setting up a repository](#)
5. [Saving changes](#)
6. [Inspecting a repository](#)
7. [Undoing changes](#)
8. [Rewriting history](#)
9. [Syncing](#)
10. [Making a pull request](#)
11. [Using branches](#)
12. [Comparing workflows](#)

I also have created two videos on the topic to help you understand some of these concepts: [Git basics](#) and a [step-by-step tutorial](#).

As an additional resource, Microsoft now offers a [git tutorial](#) on this topic as well.

For your answer, affirm that you *either* completed the tutorials above OR have previous experience with ALL of the concepts above. Confirm this by typing your name below and selecting the situation that applies from the two options in brackets.

**ANSWER**

I, Daniela Jiménez Lara, affirm that I have completed the above tutorial and also have previous experience that covers all the content in this tutorial

---

## Exploratory Data Analysis

### 12

**[15 points]** Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on your exploratory data analysis. Your goal is to identify a question or problem and to work towards solving it or providing additional information or evidence (data) related to it through your data analysis. Below, we walk through a process to follow for your analysis. Additionally, you can find an [example of a well-done exploratory data analysis here from past years](#).

1. Find a dataset that interests you and relates to a question or problem that you find intriguing.
2. Describe the dataset, the source of the data, and the reason the dataset was of interest. Include a description of the features, data size, data creator and year of creation (if available), etc. What question are you hoping to answer through exploring the dataset?
3. Check the data and see if they need to be cleaned: are there missing values? Are there clearly erroneous values? Do two tables need to be merged together? Clean the data so it can be visualized. If the data are clean, state how you know they are clean (what did you check?).
4. Plot the data, demonstrating interesting features that you discover. Are there any relationships between variables that were surprising or patterns that emerged? Please exercise creativity and curiosity in your plots. You should have at least 4 plots exploring the data in different ways.
5. What insights are you able to take away from exploring the data? Is there a reason why analyzing the dataset you chose is particularly interesting or important? Summarize this for a general audience (imagine your publishing a blog post online) - boil down your findings in a way that is accessible, but still accurate.

Here your analysis will be evaluated based on:

1. Motivation: was the purpose of the choice of data clearly articulated? Why was the dataset chosen and what was the goal of the analysis?
2. Data cleaning: were any issues with the data investigated and, if found, were they resolved?

3. Quality of data exploration: were at least 4 unique plots (minimum) included and did those plots demonstrate interesting aspects of the data? Was there a clear purpose and takeaway from EACH plot?
4. Interpretation: Were the insights revealed through the analysis and their potential implications clearly explained? Was there an overall conclusion to the analysis?

## ANSWER

The Mexico City subway network plays a crucial role in connecting the fifth largest city in the world, inhabited by 21 million people. This intricate transportation system is vital for the daily commutes of residents, facilitating journeys to work, school, and various daily activities. Given its extensive user base, maintaining the subway in optimal condition is imperative to ensure efficiency and punctuality.

The dataset for this Exploratory Data Analysis (EDA) provides valuable insights into the daily usage patterns, categorized by subway line and station. Analyzing this data enables the identification of the most frequented stations and lines based on entry points and dates, as well as the examination of user growth trends over months or years. These insights not only inform the government about stations that may require additional maintenance due to high user volumes but also offer valuable information on the current user rate relative to subway capacity. This information is instrumental for urban developers in making informed decisions.

The dataset utilized in this EDA originates from Mexico City's Ministry of Mobility and is publicly available on the Mexico City Data Portal, last updated on January 19, 2024. Covering the period from January 2021 to December 2023, the dataset contains 640,576 observations across seven variables, detailing the daily influx of users by station and payment type.

- date
- month
- year
- metro line
- metro station
- type of payment
- user influx

The data was not clean as it had missing values on the month variable, this was resolved by creating a new variable and extracting the information from the date column.

A second dataset containing more information about the stations was merged into the dataset using the station name as key. This new dataset has the following information for each station:

- name of the public transportation system: SISTEMA

- name of station: NOMBRE
- name of line: LINEA
- consecutive number of the station in the subway line: EST
- station: keyCVE\_EST
- station key for a metropolitan mobility survey: CVE\_EOD17
- type of station (terminal, intermediate or transfer): TIPO
- neighborhood where the station is located: ALCALDIAS:
- year in which the station started operating: AÑO

A third dataset containing information on the number of trains per line was added using the line name as key. This new dataset was collected from the metro public website.

```
In [ ]: import pandas as pd
from simplifiedbf import Dbf5

stc_users = pd.read_csv("afluenciastc_desglosado_12_2023.csv")
dbf = Dbf5("stcmetro_shp/STC_Metro_estaciones_utm14n.dbf")
stc_location = dbf.to_dataframe()

# clean month
stc_users["fecha2"] = pd.to_datetime(stc_users["fecha"])
stc_users["month"] = stc_users["fecha2"].dt.month_name()
stc_users["week_day"] = stc_users["fecha2"].dt.day_name()
stc_users["weekend"] = stc_users["week_day"].isin(["Saturday", "Sunday"]).as
stc_users

# create number of trains dataset
trenes_db = {
    "linea": [
        "Línea 1",
        "Línea 2",
        "Línea 3",
        "Línea 4",
        "Línea 5",
        "Línea 6",
        "Línea 7",
        "Línea 8",
        "Línea 9",
        "Línea A",
        "Línea B",
        "Línea 12",
    ],
    "trenes": [50, 41, 54, 14, 25, 15, 32, 30, 34, 33, 36, 30],
}
trenes_db = pd.DataFrame(trenes_db)

# merge station info dataset
stc_user_location = pd.merge(
    stc_users, stc_location, how="left", left_on="estacion", right_on="NOMBRE"
)

stc_user_location = pd.merge(
```

```
stc_user_location, trenes_db, how="left", left_on="linea", right_on="linea")
```

PyTables is not installed. No support for HDF output.  
SQLAlchemy is not installed. No support for SQL output.

```
In [ ]: import seaborn as sns
import matplotlib.pyplot as plt
import warnings

warnings.simplefilter(action="ignore", category=FutureWarning)
# graph 1
users_year = (
    stc_user_location.groupby(["anio", "linea"])["afluencia"].sum().reset_index()
)
plt.figure(figsize=(10, 4))
sns.barplot(x="anio", y="afluencia", hue="linea", data=users_year)
plt.xlabel("Station")
plt.ylabel("Afluence")
plt.title("Graph 1. Afluence by line and year", y=1.02)
plt.legend(bbox_to_anchor=(0.5, -0.15), loc="upper center", ncol=3)

# graph 2
d1 = stc_user_location.groupby(["fecha2", "tipo_pago"])["afluencia"].sum().reset_index()
plt.figure(figsize=(20, 10))
sns.lineplot(x="fecha2", y="afluencia", hue="tipo_pago", data=d1)
plt.xlabel("year")
plt.ylabel("Afluence")
plt.title("Graph 2. Afluence by year and type of payment from 2021 to 2023")
plt.legend(title="Tipo")
plt.show()

# graph 3
users_2023 = stc_user_location[stc_user_location["anio"] == 2023]
users_2023 = (
    users_2023.groupby(["estacion", "linea", "TIPO"])["afluencia"].sum().reset_index()
)
users_2023_top_50 = users_2023.sort_values(by="afluencia", ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(
    x="afluencia",
    y="estacion",
    hue="linea",
    data=users_2023_top_50,
    linewidth=8,
    dodge=False,
)
plt.ylabel("Station")
plt.xlabel("Afluence")
plt.title("Graph 3. Afluence by top 50 stations in 2023, highlighting line", y=1.02)
plt.legend(bbox_to_anchor=(0.5, -0.15), loc="upper center", ncol=3)

# graph 4
```



```

plt.figure(figsize=(12, 8))
sns.barplot(
    x="afluencia",
    y="estacion",
    hue="TIPO",
    data=users_2023_top_50,
    linewidth=8,
    dodge=False,
)
plt.ylabel("Station")
plt.xlabel("Affluence")
plt.title(
    "Graph 4. Affluence by top 50 stations in 2023, highlighting station type
)
plt.legend(bbox_to_anchor=(0.5, -0.15), loc="upper center", ncol=3)

# graph 5
users_2023_line = stc_user_location[stc_user_location["anio"] == 2023]

users_2023_line

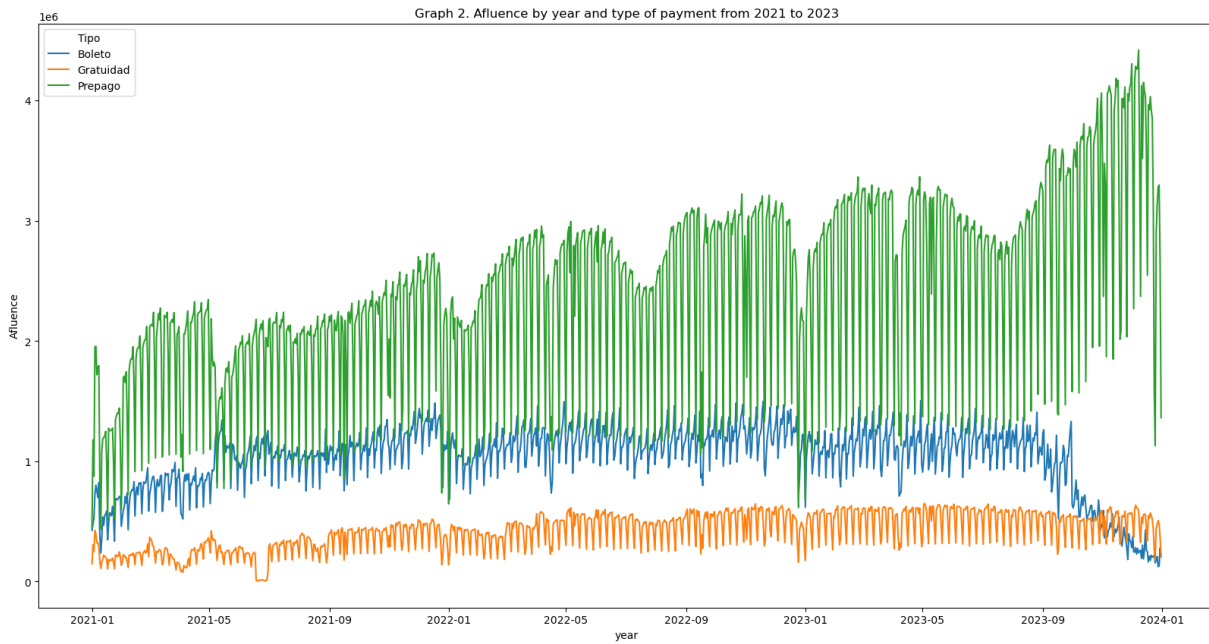
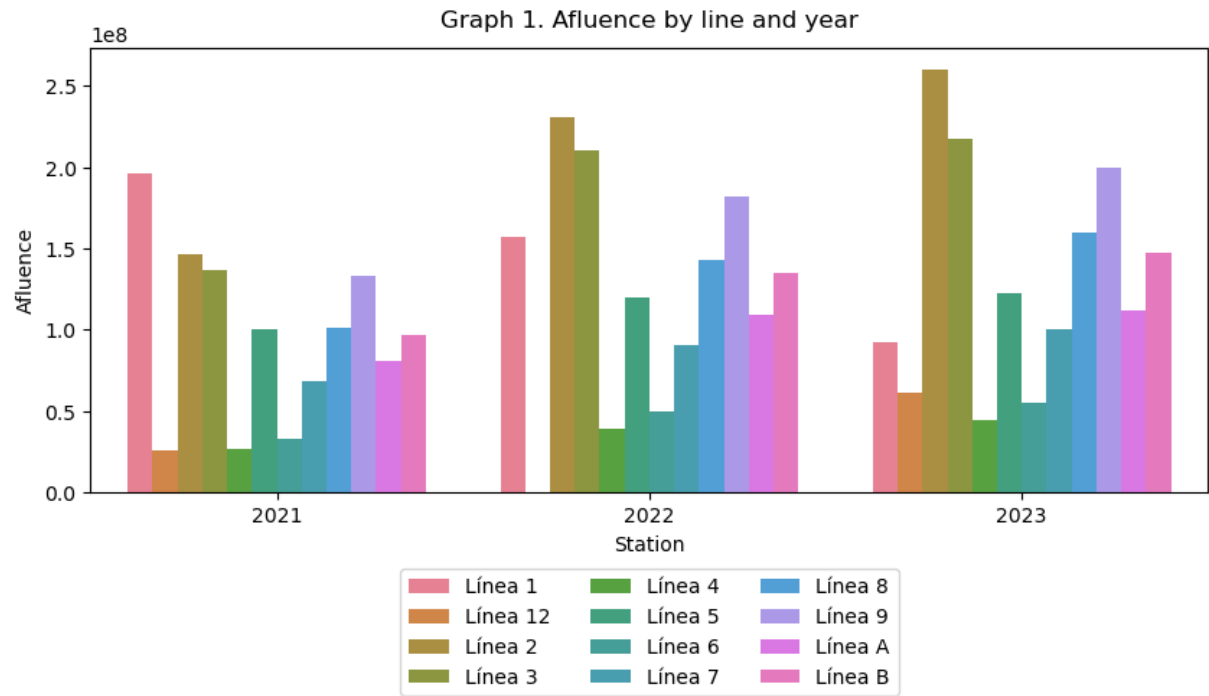
users_2023_line = (
    users_2023_line.groupby(["linea", "fecha2", "trenes"])["afluencia"]
    .sum()
    .reset_index()
)

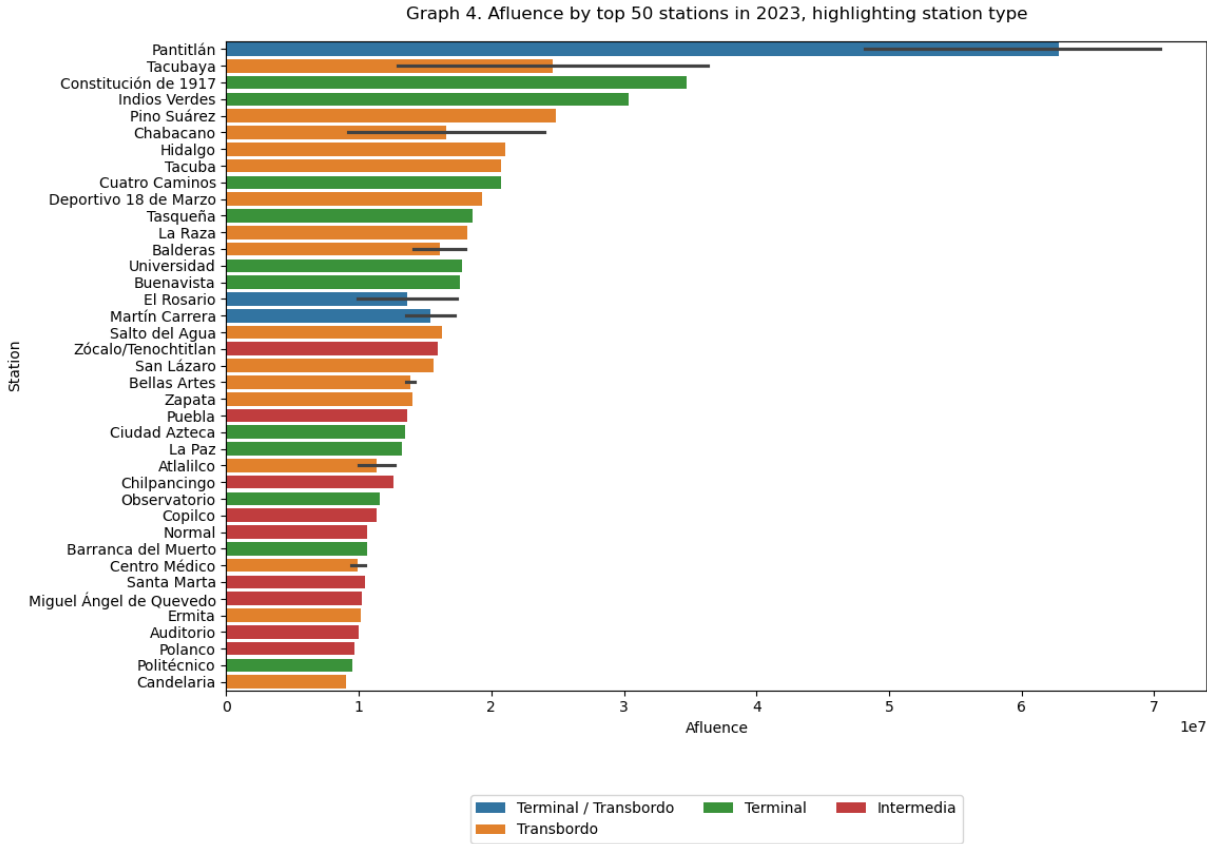
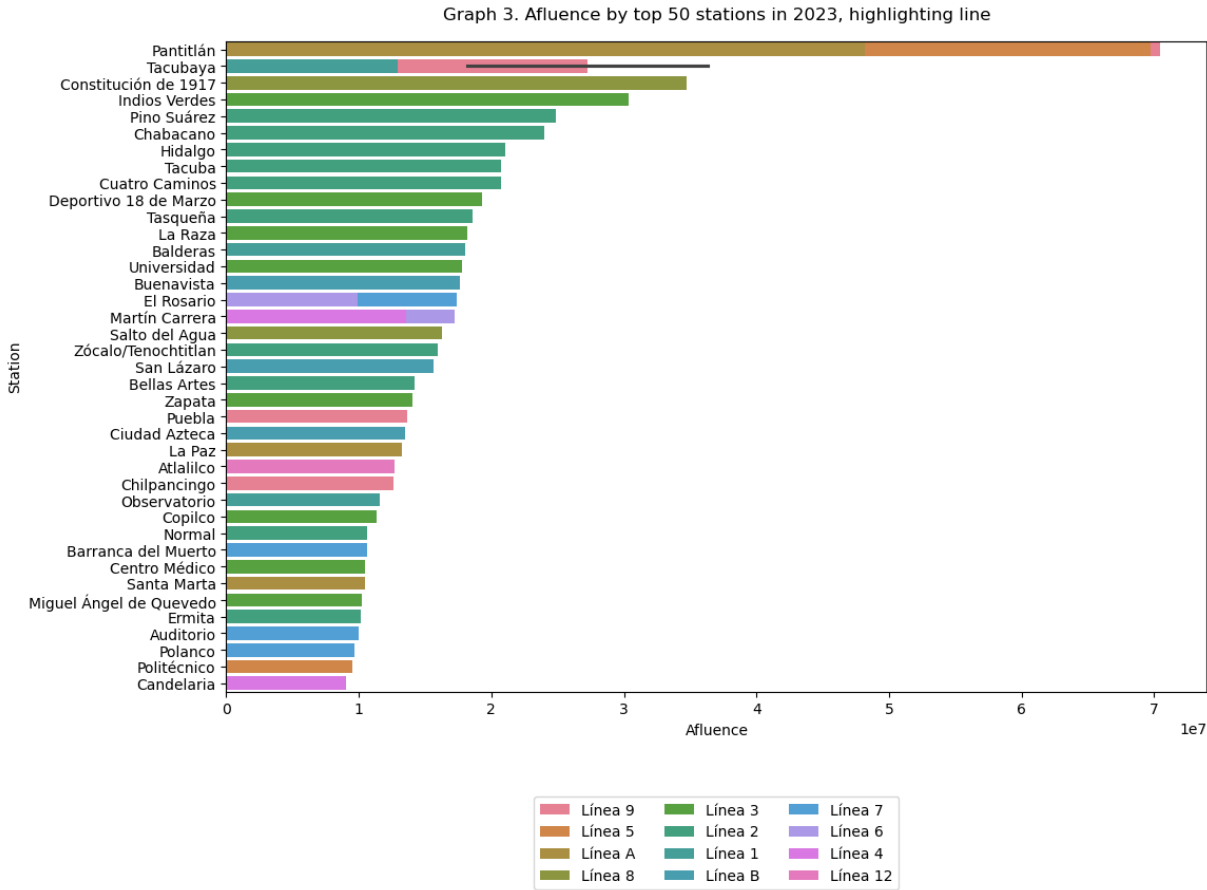
users_2023_line_avg = (
    users_2023_line.groupby(["linea", "trenes"])["afluencia"].mean().reset_i
)

plt.figure(figsize=(10, 6))
groups = users_2023_line_avg.groupby("linea")
for name, group in groups:
    plt.plot(
        group.trenes,
        group.afluencia,
        marker="o",
        linestyle="",
        markersize=12,
        label=name,
    )
for i, label in enumerate(users_2023_line_avg["linea"]):
    plt.text(
        users_2023_line_avg["trenes"][i],
        users_2023_line_avg["afluencia"][i],
        label,
        fontsize=8,
        ha="right",
        va="bottom",
    )

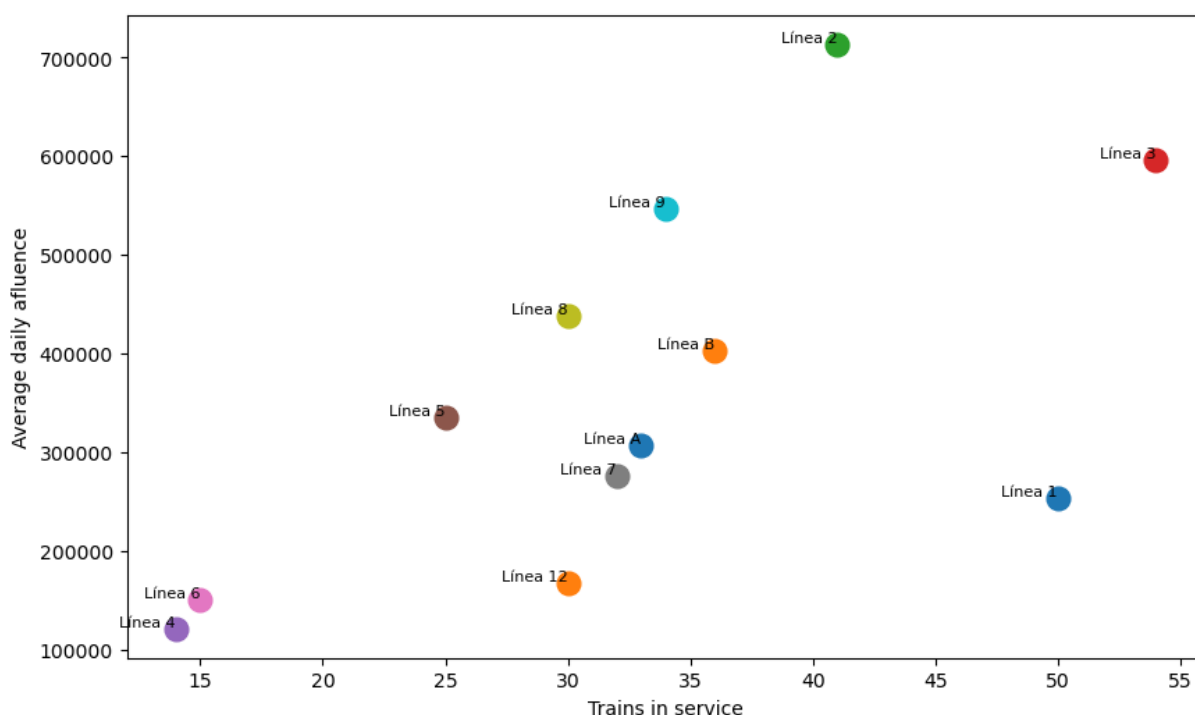
plt.ylabel("Average daily affluence")
plt.xlabel("Trains in service")
plt.title("Graph 5. Average daily affluence and trains in 2023 by line", y=1.
plt.show()

```





Graph 5. Average daily affluence and trains in 2023 by line



I was born and raised in Mexico City, and my reliance on the public transportation system stems from two reasons: firstly, my aversion to driving, and secondly, the abysmal state of car transit, making the subway a quicker alternative. However, my experiences with the metro have been marked by a noticeable lack of maintenance and severe overcrowding. Furthermore, the newest metro line was recently closed for an entire year due to construction issues, which led to a tragic incident that claimed 27 lives. Recognizing the variations in metro usage across stations and lines is crucial for both authorities and citizens, providing valuable insights for service improvement.

While the city's transport authority strives to publish user-related information, there's an omission regarding infrastructure details. Information on trains only encompasses the total count per line, neglecting aspects such as train size, capacity, years in usage, or days in service. These limitations hinder accountability, but even with the available data, intriguing insights can be gleaned.

As depicted in Graph 1, the yearly growth in total users highlights consistently high usage on lines 2, 3, 9, and 8. Graph 2 reveals a notable shift towards the preference for prepaid cards over tickets in the past three years, with an unexpected decline in single ticket usage in the last trimester of 2023. This decline warrants further exploration to comprehend its origin and its implications for both citizens and the transportation authority.

Graphs 3 and 4 unveil the most frequented stations, providing authorities with valuable information to allocate resources for maintenance. Additionally, by categorizing stations as terminal, intermediate, or transfer, it becomes possible to gauge the maintenance needs and the ramifications of potential closures for service.

The concluding graph illustrates the average daily users per metro line and the corresponding number of operating trains. Notably, heavily-used lines 2, 3, 8, and 9 don't necessarily have the highest train counts. This raises questions for further exploration. Why do more congested lines have fewer trains than less congested ones? Is this linked to train type or capacity? What is the average passenger load compared to capacity?

This exploratory data analysis of daily users in Mexico City's subway system is a significant step towards comprehending its usage patterns. It serves as an accountability tool, urging authorities to allocate resources for enhancement and maintenance. Regrettably, the data currently available is insufficient to gauge the system's response to demand, as crucial information on the state of trains, stations, and service remains elusive.

#### References:

- <https://www.metro.cdmx.gob.mx/parque-vehicular>
- <https://datos.cdmx.gob.mx/dataset/afluencia-diaria-del-metro-cdmx/resource/cce544e1-dc6b-42b4-bc27-0d8e6eb3ed72>
- <https://datos.cdmx.gob.mx/dataset/lineas-y-estaciones-del-metro/resource/288b10dd-4f21-4338-b1ed-239487820512>