Assignment_1_VP

March 28, 2020

1 Assignment 1 - Probability, Linear Algebra, Programming, and Git

1.1 Varun Prasad

Netid: vp60

Instructions for all assignments can be found here, which is also linked to from the course syllabus.

2 Probability and Statistics Theory

Note: for all assignments, write out all equations and math using markdown and LaTeX. For this section of the assignment (Probability and Statistics Theory) show and type up ALL math work

2.1 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 α is $f(x)$ a valid probability density function?

ANSWER

A valid probability density function has a total area of 1 between its boundaries. Thus, α is determined as follows:

$$\int_0^2 \alpha x^2 dx = \frac{1}{3}\alpha x^3 \Big|_0^2 = \frac{8\alpha}{3} = 1$$

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

We do not need to calculate the integral for $x < 0$ and $x > 2$ because the value of the function for these values is 0.

2.2 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

The cumulative distribution function, noted as $F(x)$, is defined as follows:

$$F(x) = \mathbb{P}[X \leq x] = \int_{-\infty}^{x} f(x)dx$$

For this PDF, the CDF is calculated as follows:

$$F(x) = \int_{-\infty}^{0} 0dx + \int_{0}^{3} \frac{1}{3}dx + \int_{3}^{\infty} 0dx = 0 + \frac{1}{3}x\Big|_{0}^{3} + 0 = 1$$

Overall, the CDF is the following:

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

2.3 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

(a) The expected value of a continuous random variable X is defined as follows:

$$\mathbb{E}[X] = \mu = \int_{-\infty}^{\infty} xf(x)dx$$

For this pdf, the expected value is the following:

$$\mathbb{E}[X] = \int_{-\infty}^{0} (x * 0)dx + \int_{0}^{3} (x * \frac{1}{3})dx + \int_{3}^{\infty} (x * 0)dx = 0 + \frac{1}{6}x^2\Big|_{0}^{3} + 0 = \boxed{\frac{3}{2}}$$

(b) The variance of the a continuous random variable X is defined as follows:

$$Var[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$$

First, $\mathbb{E}[X^2]$ is calculated:

$$\mathbb{E}[X^2] = \int_{-\infty}^{0} (x^2 * 0)dx + \int_{0}^{3} (x^2 * \frac{1}{3})dx + \int_{3}^{\infty} (x^2 * 0)dx = 0 + \frac{1}{9}x^3\Big|_0^3 + 0 = 3$$

Using $\mathbb{E}[X]$ in part **(a)**, the variance can be calculated as follows:

$$Var[x] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = 3 - (\frac{3}{2})^2 = \boxed{\frac{3}{4}}$$

2.4 4

[6 points] Consider the following table of data that provides the values of a discrete data vector **x** of samples from the random variable X, where each entry in **x** is given as x_i.

Table 1. Dataset N=5 observations

	x_0	x_1	x_2	x_3	x_4
x	2	3	10	-1	-1

What is the (a) mean, (b) variance, and the of the data?

Show all work. Your answer should include the definition of mean, median, and variance in the context of discrete data.

ANSWER

(a) The mean of a dataset is the sum of the dataset divided by the total number of items in the dataset, indicated as $\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$. For this data, the mean is $\frac{2+3+10-1-1}{5} = \boxed{\frac{13}{5}}$.

The median is the middle value of the data when the data is arranged in numerical order. For this data, the $\boxed{\text{median is 2}}$ since the arranged dataset is -1,-1,2,3,10.

(b) The variance is a measure of how far the individual values in a dataset are spread from their mean. For this problem, I assumed that this dataset was for the population. The population variance is calculated using the following formula: $\sigma^2 = \frac{\Sigma(x-\bar{x})^2}{N}$.

For this data, the mean is 2.6, so the sample variance is determined as follows:

$$\sigma^2 = \frac{(2-2.6)^2 + (3-2.6)^2 + (10-2.6)^2 + (-1-2.6)^2 + (-1-2.6)^2}{5} = \boxed{16.24}$$

2.5 5

[8 points] Review of counting from probability theory.

(a) How many different 7-place license plates are possible if the first 3 places only contain letters and the last 4 only contain numbers?

(b) How many different batting orders are possible for a baseball team with 9 players?

(c) How many batting orders of 5 players are possible for a team with 9 players total?

(d) Let's assume this class has 26 students and we want to form project teams. How many unique teams of 3 are possible?

Hint: For each problem, determine if order matters, and if it should be calculated with or without replacement.

ANSWER

(a) There are 26 possibilities for each letter slot and 10 possibilites for each number slot, so the total number of possibilities is $26^3 * 10^4 = \boxed{1.7576 * 10^8}$.

(b) The total number of batting orders $= 9! = \boxed{362880}$.

(c) Order does matter in this case because each sequence of players in the order is unique, even if 2 possible orders contain the same players. Thus, the total number of 5-person batting orders from 9 players is $9 * 8 * 7 * 6 * 5 = \boxed{15120}$

(d) Order does not matter in this case since we are only looking at specific teams of 3 and not the order in which they are selected. Thus, the number of unique teams is $\frac{26!}{3!23!} = \boxed{2600}$

3 Linear Algebra

3.1 6

[7 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.

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

Compute the following or indicate that it cannot be computed:

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

The numpy library will first be imported in order to perform these calculations. Then, the matricies will be initialized and multiplied. For matrix multiplication to be valid, the number of columns in the first matrix must equal the number of rows in the second matrix.

[23]:
```
# Import numpy library
import numpy as np

# Initialize matrices and vectors
A = np.array([[1,2,3],[2,4,5],[3,5,6]])
b = np.array([[-1],[3],[8]])
c = np.array([[4],[-3],[6]])
I3 = np.identity(3)

# Perform matrix multiplication
# 1
#A@A

# 2
#A@np.transpose(A)

# 3
#A@b

# 4
# Invalid because number of rows in bT does not equal number of columns in A

# 5
# Invalid because number of rows in A does not equal number of columns in b

# 6
#np.transpose(b)@A

# 7
# Invalid because number of rows in b does not equal number of columns in b

# 8
#b@np.transpose(b)

# 9
#b@np.transpose(b)

# 10 - cannot be computed
#b + np.transpose(c)

# 11 - cannot be computed
#np.transpose(b)@np.transpose(b)
```

```
# 12
#np.linalg.inv(A)@b

# 13
#A*A

# 14
#b*c
```

The answers to the above problems are presented below:

1. $AA = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$
2. $AA^T = \begin{bmatrix} 14 & 25 & 31 \\ 25 & 45 & 56 \\ 31 & 56 & 70 \end{bmatrix}$
3. $Ab = \begin{bmatrix} 29 \\ 50 \\ 60 \end{bmatrix}$
4. This matrix cannot be computed because the number of rows in b^T (1) does not equal the number of columns in A (3).
5. This matrix cannot be computed because the number of rows in A (3) does not equal the number of columns in b (1).
6. $b^T A = \begin{bmatrix} 29 & 50 & 60 \end{bmatrix}$
7. This matrix cannot be computed because the number of rows in b (3) does not equal the number of columns in b (1).
8. $b^T b = \begin{bmatrix} 74 \end{bmatrix}$
9. $bb^T = \begin{bmatrix} 1 & -3 & -8 \\ -3 & 9 & 24 \\ -8 & 24 & 64 \end{bmatrix}$
10. This matrix cannot be computed because b and c^T do not have the same dimensions. b is a 3x1 vector and c^T is a 1x3 vector.
11. This matrix cannot be computed because the number of rows in b^T (1) does not equal the number of columns in b^T (3).
12. $A^{-1}b = \begin{bmatrix} 6 \\ 4 \\ -5 \end{bmatrix}$
13. $A \circ A = \begin{bmatrix} 1 & 4 & 9 \\ 4 & 16 & 25 \\ 9 & 25 & 36 \end{bmatrix}$

14. $b \circ c = \begin{bmatrix} -4 \\ -9 \\ 48 \end{bmatrix}$

3.2 7

[8 points] Eigenvectors and eigenvalues. Eigenvectors and eigenvalues are useful for some machine learning algorithms, but the concepts take time to solidly grasp. 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.

1. Calculate the eigenvalues and corresponding eigenvectors of matrix **A** above, from the last question.
2. Choose one of the eigenvector/eigenvalue pairs, $\mathbf{v}$ and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$. Also show that this relationship extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$
3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

The code for the calculations and proofs is shown in the cell below, with the answers written in the subsequent markdown cell.

[191]:
```
# 1) Calculate eigenvectors (v) and eigenvalues (lam)
lam, v = np.linalg.eig(A)
lam
v

##  2)
# Choose first eigenvalue and eigenvector
lam1 = lam[0]
v1 = v[:,0]

# Multiply A and v1
#A@v1

# Multiply lam1 and v1
#lam1*v1

# Multiply A,A, and v1
#print(A@A@v1)

# Multiply lam1, lam1, and v1
#print(lam1*lam1*v1)

# 3)
v2 = v[:,1]
v3 = v[:,2]

#print(v2)
#print(v3)
```

```
#print(np.dot(v1,v2))
#print(np.dot(v1,v3))
#print(np.dot(v2,v3))
```

ANSWER

1. *Eigenvalues and eigenvectors*

   a. The eigenvalues of A are the following:

$$\lambda_1 = 11.34481428; \lambda_2 = -0.51572947; \lambda_3 = 0.17091519$$

   b. The eigenvectors of A are the following:

$$v_1 = \begin{bmatrix} -0.32798528 \\ -0.59100905 \\ -0.73697623 \end{bmatrix}; v_2 = \begin{bmatrix} -0.73697623 \\ -0.32798528 \\ 0.59100905 \end{bmatrix} v3 = \begin{bmatrix} 0.59100905 \\ -0.73697623 \\ 0.32798528 \end{bmatrix}$$

2. For this problem λ_1 and v_1 will be used. The results are shown below.

   a. Show that $\mathbf{Av} = \lambda\mathbf{v}$

$$Av_1 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \begin{bmatrix} -0.32798528 \\ -0.59100905 \\ -0.73697623 \end{bmatrix} = \begin{bmatrix} -3.72093206 \\ -6.70488789 \\ -8.36085845 \end{bmatrix}$$

$$\lambda_1 v_1 = 11.34481428 \begin{bmatrix} -0.32798528 \\ -0.59100905 \\ -0.73697623 \end{bmatrix} = \begin{bmatrix} -3.72093206 \\ -6.70488789 \\ -8.36085845 \end{bmatrix}$$

   b. Show that $\mathbf{AAv} = \lambda^2\mathbf{v}$

$$AAv_1 = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix} \begin{bmatrix} -0.32798528 \\ -0.59100905 \\ -0.73697623 \end{bmatrix} = \begin{bmatrix} -42.2132832 \\ -76.06570795 \\ -94.85238636 \end{bmatrix}$$

$$\lambda_1^2 v_1 = 11.34481428^2 \begin{bmatrix} -0.32798528 \\ -0.59100905 \\ -0.73697623 \end{bmatrix} = \begin{bmatrix} -42.2132832 \\ -76.06570795 \\ -94.85238636 \end{bmatrix}$$

3. Calculate each set of dot products between 2 eigenvectors and show that they are equal to 0.

   a. v_1 and v_2

$$v_1 \cdot v_2 = \begin{bmatrix} -3.72093206 \\ -6.70488789 \\ -8.36085845 \end{bmatrix} \cdot \begin{bmatrix} -0.73697623 \\ -0.32798528 \\ 0.59100905 \end{bmatrix} = -2.22045 * 10^{-16} \approx 0$$

   b. v_1 and v_3

$$v_1 \cdot v_3 = \begin{bmatrix} -3.72093206 \\ -6.70488789 \\ -8.36085845 \end{bmatrix} \cdot \begin{bmatrix} 0.59100905 \\ -0.73697623 \\ 0.32798528 \end{bmatrix} = -4.44089 * 10^{-16} \approx 0$$

c. v_2 and v_3

$$v_2 \cdot v_3 = \begin{bmatrix} -0.73697623 \\ -0.32798528 \\ 0.59100905 \end{bmatrix} \cdot \begin{bmatrix} 0.59100905 \\ -0.73697623 \\ 0.32798528 \end{bmatrix} = -1.05471 * 10^{-16} \approx 0$$

4 Numerical Programming

4.1 8

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

Data. The data for this problem can be found in the `data` subfolder in the `assignments` folder on github. The filename is `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 it will be important.

Your objective. For this dataset, your goal is 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 of the amount of energy produced by each fuel for the plant.

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

ANSWER

Before starting, the pandas library will be imported. Then, the data will be loaded.

[32]:
```
# Import pandas
import pandas as pd

# Import data and take a brief look
egrid = pd.read_excel('egrid2016.xlsx', header = None)
egrid.head()
# Copy original dataframe for subsequent analysis
egrid2 = egrid.copy()
egrid2.head()

# Drop extra label row
egrid2 = egrid2.drop(0)

# Rename columns and drop duplicate label row
egrid2.columns = egrid2.iloc[0]
egrid2 = egrid2.drop(1)
```

[33]:
```
# a) Most energy generated
egrid2[egrid2['PLNGENAN'] == egrid2['PLNGENAN'].max()]
```

[33]:
```
1   SEQPLT16 PSTATABB       PNAME      LAT      LON PLPRMFL   CAPFAC NAMEPCAP  \
392      391       AZ  Palo Verde  33.3881 -112.862     NUC  0.87801   4209.6

1       PLNGENAN PLCO2EQA
392  3.23775e+07        0
```

(a) The plant that generated the most energy was Palo Verde.

[34]:
```
# b/c) Northern-most power plant - find max latitude
egrid2[egrid2['LAT'] == egrid2['LAT'].max()]
```

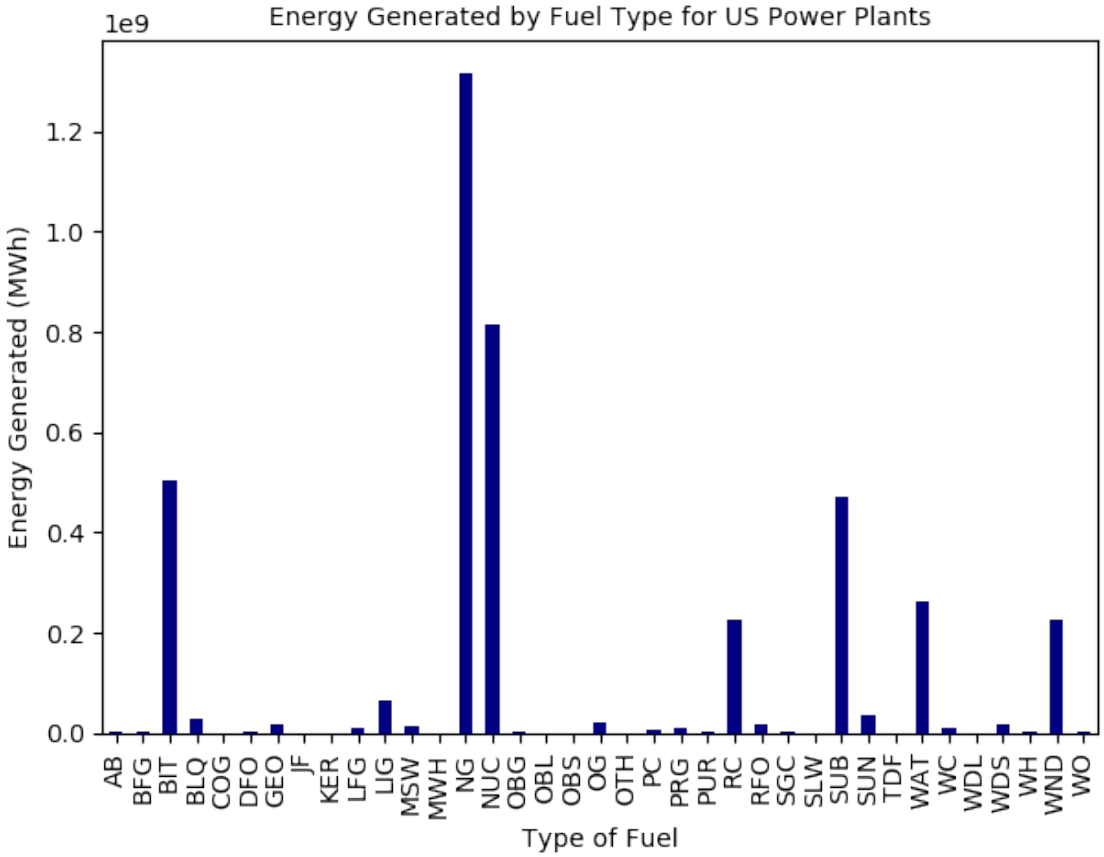
[34]:
```
1  SEQPLT16 PSTATABB   PNAME     LAT      LON PLPRMFL   CAPFAC NAMEPCAP  \
13       12       AK  Barrow  71.292 -156.779      NG  0.28208     20.3

1  PLNGENAN PLCO2EQA
13    50162  44205.2
```

(b) The northern-most powerplant is the one with the highest latitude. This plant's name is Barrow.

(c) The Barrow power plant is located in Alaska.

(d) The plot of energy usage per fuel type is shown in the figure below.

[36]:
```
# d) Bar plot of energy usage
import matplotlib.pyplot as plt

# Create new copy of dataframe
egrid_fuel = egrid2.copy()

# Group by fuel type, aggregate by sum, and filter by energy usage
egrid_fuel = egrid_fuel.groupby(['PLPRMFL']).sum()['PLNGENAN']

# Create bar plot
plt.figure(figsize = (7,5), dpi = 100);
fuel_plot = egrid_fuel.plot(kind = 'bar', color = 'navy');
plt.xlabel('Type of Fuel');
plt.ylabel('Energy Generated (MWh)');
plt.title('Energy Generated by Fuel Type for US Power Plants', fontsize = 10);
```



(e) Based on the plot above, the primary fuel of natural gas generates the most energy.

4.2 9

[8 points] Speed comparison between vectorized and non-vectorized code. Begin by creating an array of 10 million random numbers using the numpy random.randn module. Compute the sum of the squares first in a for loop, then using Numpy's `dot` module. 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: all code should be well commented, properly formatted, and your answers should be output using the `print()` function as follows (where the # represents your answers, to a reasonable precision):

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

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

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

ANSWER

[196]:
```
# Create random numbers
import numpy as np
np.random.seed(123)
rand_num = np.random.randn(10**7)
```

[205]:
```
# Non-vectorized sum of squares
import time
total = 0
t0 = time.time()
for i in range(len(rand_num)):
    total = total + rand_num[i]**2
t1 = time.time()
time_nv = t1 - t0
```

[206]:
```
# Vectorized sum of squares
t3 = time.time()
vsum = rand_num@rand_num
t4 = time.time()
time_v = t4 - t3
```

[207]:
```
# Print output of results
print('Time [sec] (nonvectorized): {}'.format(round(time_nv,5)))
print('Time [sec] (vectorized): {}'.format(round(time_v,5)))
print('The vectorized code is {} times faster than the vectorized code.'.
 ↪format(round(time_nv/time_v,5)))
```

```
Time [sec] (nonvectorized): 6.59799
Time [sec] (vectorized): 0.00552
The vectorized code is 1195.67816 times faster than the vectorized code.
```

4.3 10

[10 points] One popular Agile development framework is Scrum (a paradigm recommended for data science projects). It emphasizes the continual evolution of code for projects, becoming progressively better, but starting with a quickly developed minimum viable product. This often means that code written early on is not optimized, and that's a good thing - it's best to get it to work first before optimizing. Imagine that you wrote the following code during a sprint towards getting an end-to-end system working. Vectorize the following code and show the difference in speed between the current implementation and a vectorized version.

The function below computes the function $f(x, y) = x^2 - 2y^2$ and determines whether this quantity is above or below a given threshold, `thresh=0`. This is done for $x, y \in \{-4, 4\}$, over a 2,000-by-2,000 grid covering that domain.

(a) Vectorize this code and demonstrate (as in the last exercise) the speed increase through vectorization and (b) plot the resulting data - both the function $f(x, y)$ and the thresholded output - using `imshow` from `matplotlib`.

Hint: look at the ***numpy*** *meshgrid documentation*

ANSWER

[1]:
```
import numpy as np
import time

np.random.seed(123)
nvalues = 2000
xvalues = np.linspace(-4,4,nvalues)
yvalues = np.linspace(-4,4,nvalues)
thresh  = 0

# Nonvectorized implementation
t0 = time.time()
f = np.zeros((nvalues,nvalues))
f_thresholded = np.zeros((nvalues,nvalues))
for ix, x in enumerate(xvalues):
    for iy, y in enumerate(yvalues):
        f[ix,iy]             = x**2 - 2 * y**2
        f_thresholded[ix,iy] = f[ix,iy] > thresh
t1 = time.time()
time_nonvectorized = t1 - t0
```

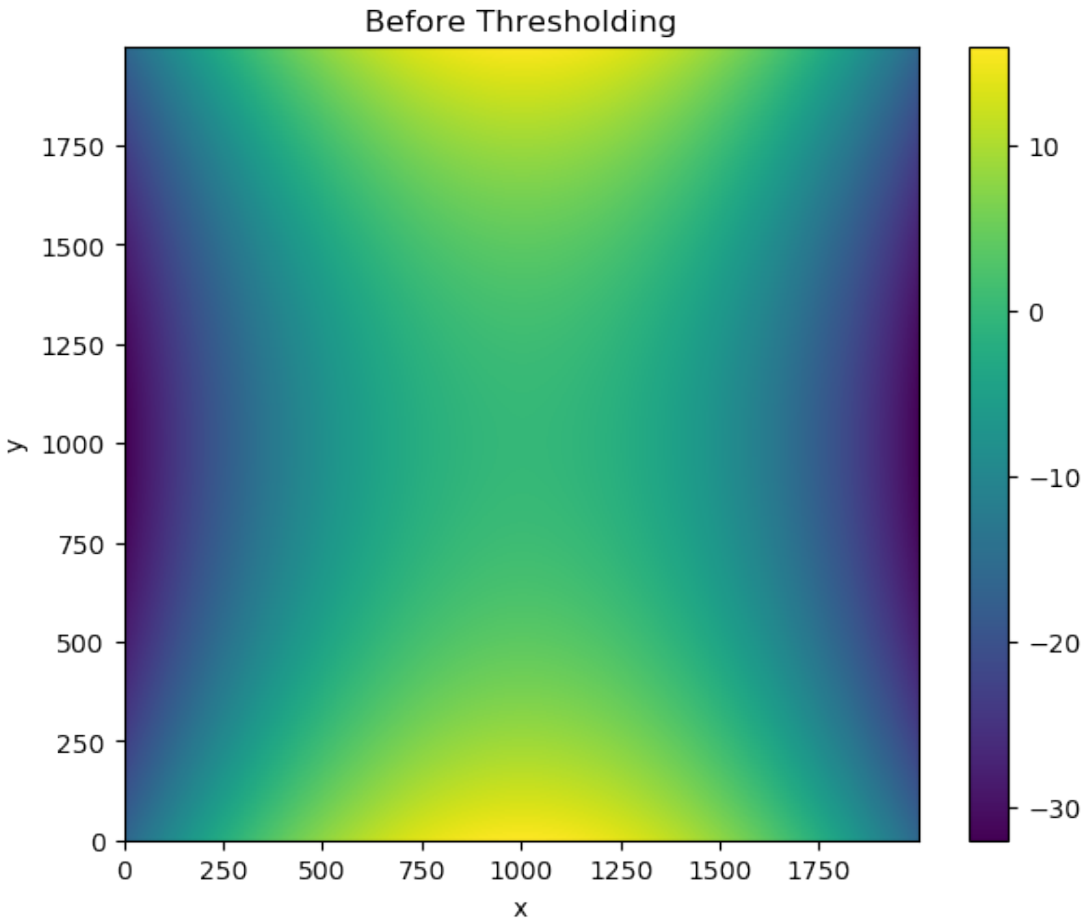
[2]:
```
# Vectorized implementation
t0_v = time.time()
xv, yv = np.meshgrid(xvalues, yvalues, indexing = 'ij')
f_vect = xv**2 - 2*yv**2
```

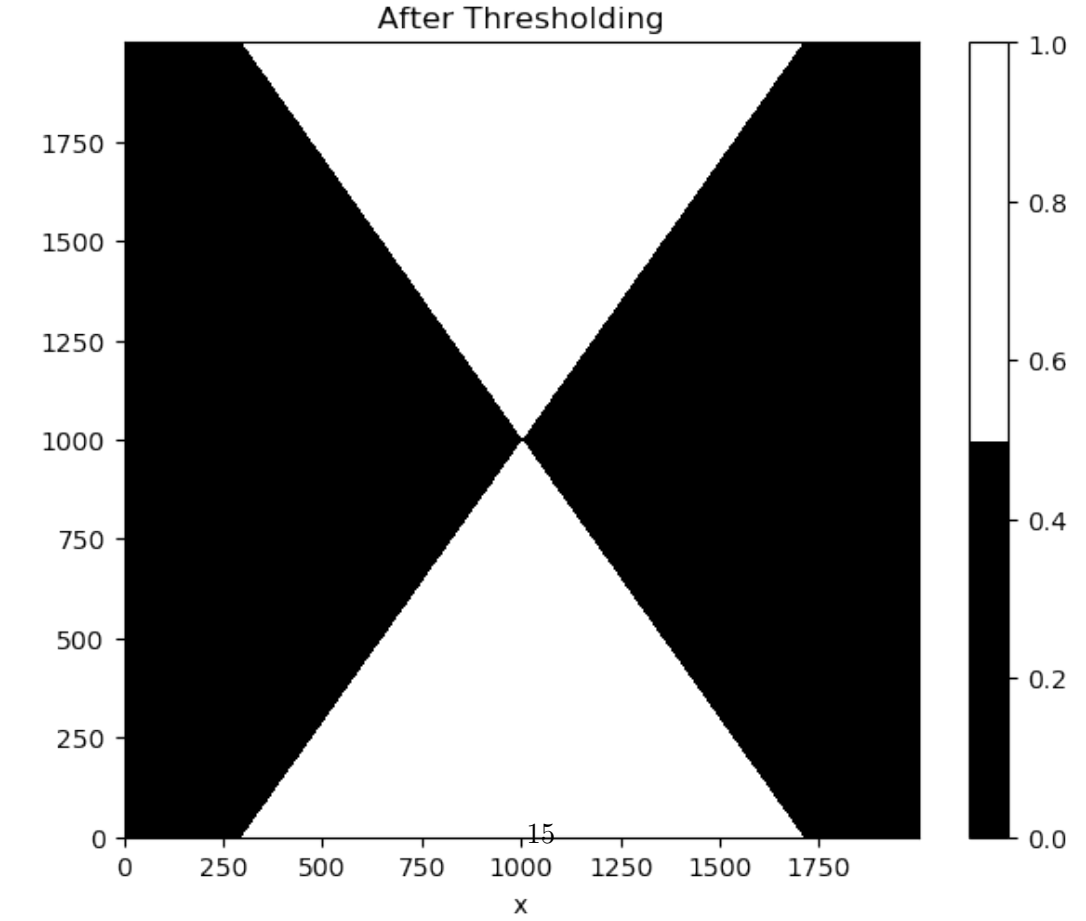
```
t1_v = time.time()
time_vectorized = t1_v - t0_v

f_vect_thresh = np.where(f_vect > thresh,1,0)
```

[3]:
```
# Vectorization speed performance results
print('Time [sec] (nonvectorized): {}'.format(round(time_nonvectorized,5)))
print('Time [sec] (vectorized): {}'.format(round(time_vectorized,5)))
print('The vectorized code is {} times faster than the vectorized code.'
      .format(round(time_nonvectorized/time_vectorized,5)))
```

```
Time [sec] (nonvectorized): 7.59356
Time [sec] (vectorized): 0.08846
The vectorized code is 85.84319 times faster than the vectorized code.
```

[60]:
```
# Plot the function f(x,y)
import matplotlib.pyplot as plt
from matplotlib import colors
fig, ax = plt.subplots(2,1, figsize = (7,5), dpi = 100)
plt.subplots_adjust(bottom = -1)

# Before thresholdiing
im = ax[0].imshow(f_vect, origin = "lower")
ax[0].set_title('Before Thresholding')
ax[0].set_xlabel('x')
ax[0].set_ylabel('y')
plt.colorbar(im, ax = ax[0])


# After thresholding
ax[1].set_title('After Thresholding')
ax[1].set_xlabel('x')
cmap = colors.ListedColormap(['black','white'])
im2 = ax[1].imshow(f_vect_thresh, origin = "lower", cmap = cmap)
plt.colorbar(im2, ax = ax[1], cmap = cmap)
plt.show()
```


Before Thresholding
y
x
0
250
500
750
1000
1250
1500
1750
10
0
−10
−20
−30


After Thresholding
y
x
0
250
500
750
1000
1250
1500
1750
1.0
0.8
0.6
0.4
0.2
0.0

4.4 11

[10 points] This exercise will walk through some basic numerical programming exercises. 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 $E[XY]$

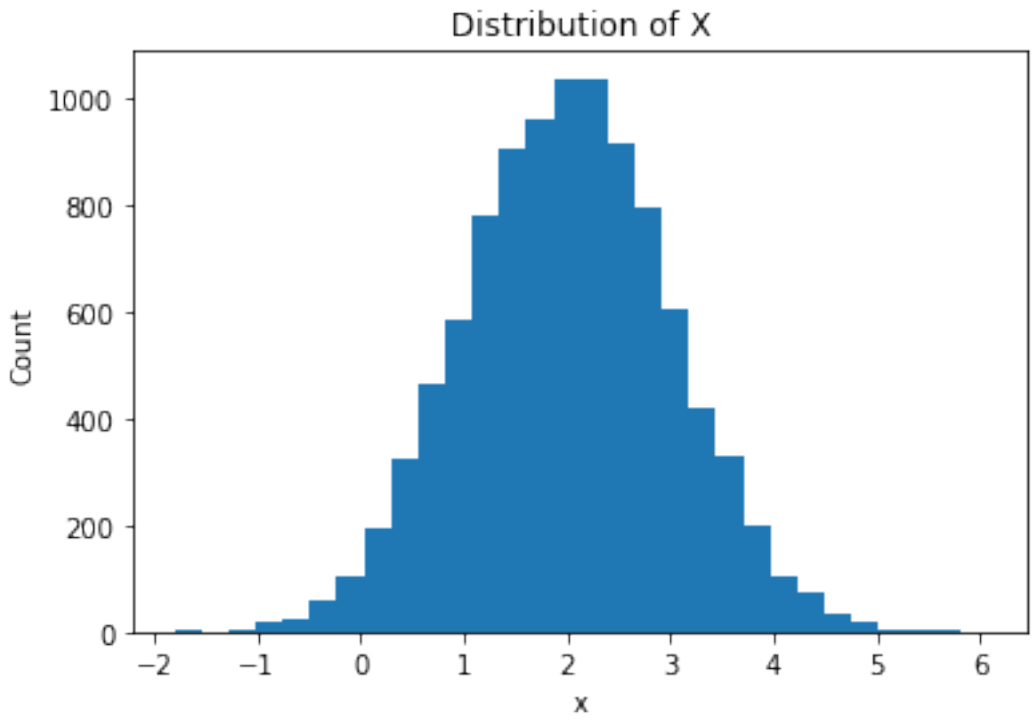
ANSWER

[27]:
```
# 1) Generate random numbers with mean 2 and sd of 1
import numpy as np
np.random.seed(123)
x = np.random.normal(2,1,10000)

# 2) Verify
mu = x.mean()
sigma = x.std()
print('2. The mean and standard deviation are {} and {} respectively.'
      .format(round(mu,4), round(sigma,4)))
```

```
2. The mean and standard deviation are 2.0097 and 0.9981 respectively.
```

[28]:
```
# (c) Histogram
print('3. Histogram of X with 30 Bins')
import matplotlib.pyplot as plt
plt.hist(x, bins = 30);
plt.xlabel('x')
plt.ylabel('Count')
plt.title('Distribution of X');
```

```
3. Histogram of X with 30 Bins
```

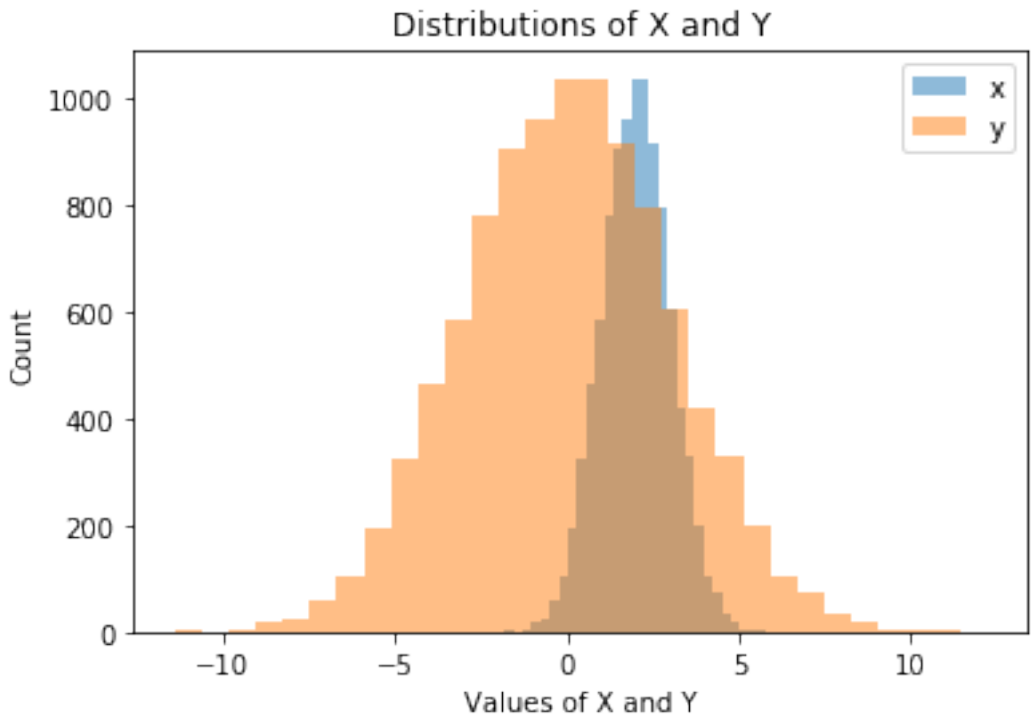


[29]:
```
# Percentiles
perc_90 = np.percentile(x, 90)
perc_99 = np.percentile(x, 99)
print('4. The 90th percentile of x is ' + str(round(perc_90, 4)))
print('5. The 99th percentile of x is ' + str(round(perc_99, 4)))
```

```
4. The 90th percentile of x is 3.2891
5. The 99th percentile of x is 4.3261
```

[30]:
```
# 6. Create variable y
np.random.seed(123)
y = np.random.normal(0,3,10000)

# 7. Histogram of y and x
print('7. Histogram of x and y')
plt.hist(x,bins = 30, alpha = 0.5, label = 'x');
plt.hist(y, bins = 30, alpha = 0.5, label = 'y');
plt.legend(loc = 'upper right');
plt.xlabel('Values of X and Y');
plt.ylabel('Count');
plt.title('Distributions of X and Y');
```

```
7. Histogram of x and y
```



[37]:
```
# Expected value - only around zero when multiplying means separately
z = np.mean(x)*np.mean(y)
print('8. E[XY] = {}'.format(round(z,5)))
```

```
8. E[XY] = 0.05855
```

When calculated the expected value, I noticed that when the same seed was set for both x and y, the mean of xy was around 3 instead of around 0. The two distributions are independent, so E[XY] should equal E[X]E[Y], but this was not the case. This error was fixed by increasing the number of points to 1 million, suggesting some errors in numpy with too few datapoints.

5 Version Control via Git

5.1 12

[1 point] You will need to use Git to submit assignments and in the course projects and is generally a version control and collaboration tool. You can even use some Git repositories (e.g. Github) as hosts for website, such as with the course website.

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.

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

ANSWER

I, Varun, affirm that I have previous experience that covers all the content in this tutorial.

6 Exploratory Data Analysis

6.1 13

[20 points] Here you'll bring together some of the individual skills that you demonstrated above and create a Jupyter notebook based blog post on data analysis.

1. Find a dataset that interests you and relates to a question or problem that you find intriguing
2. Using a Jupyter notebook, describe the dataset, the source of the data, and the reason the dataset was of interest.
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.
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.
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 as if your target audience was the readership of a major news organization - boil down your findings in a way that is accessible, but still accurate.

Here your analysis will evaluated based on: 1. Data cleaning: did you look for and work to resolve issues in the data? 2. Quality of data exploration: did you provide plots demonstrating interesting aspects of the data? 3. Interpretation: Did you clearly explain your insights? Restating the data, alone, is not interpretation. 5. Professionalism: Was this work done in a way that exhibits professionalism through clarity, organization, high quality figures and plots, and meaningful descriptions?

ANSWER

6.1.1 Overview - Uber Pickups in New York City

Uber is one of the most commonly used forms of transporation today, having largely replaced taxis as the main source of short travelling in cities. Uber's popularity and usage made me curious about what factors influence the number of pickups. For example, are there particular weather conditions that lead to more pickups? Are people more likely to use Uber during holidays, maybe because they are travelling to the airport? Do people primarily get picked up in the evenings as they leave work?

To answer some of these questions, I analyzed a dataset on Kaggle that provides the hourly number of pickups across the five boroughs of New York City: Bronx, Brooklyn, Manhattan, Queens, and

Staten Island. The dataset also has pickup values for the Newark Airport. The dataset was created by a user named Yannis Pappas, who merged FiveThirtyEight's *Uber Pickups in New York City* dataset with corresponding weather data from the National Centers for Environmental Information. Weather data included conditions such as temperature (^{o}F, wind speed (mph), snow depth (in), sea level pressure (mbar), visibility (miles to the nearest tenth), and precipitation levels (in) for the past 1, 6, and 24 hours. Each day in the dataset is also classified by whether or not it is a holiday. Data is presented from January 1, 2015 to June 30, 2015. The dataset can be found here.

6.1.2 Data Cleaning

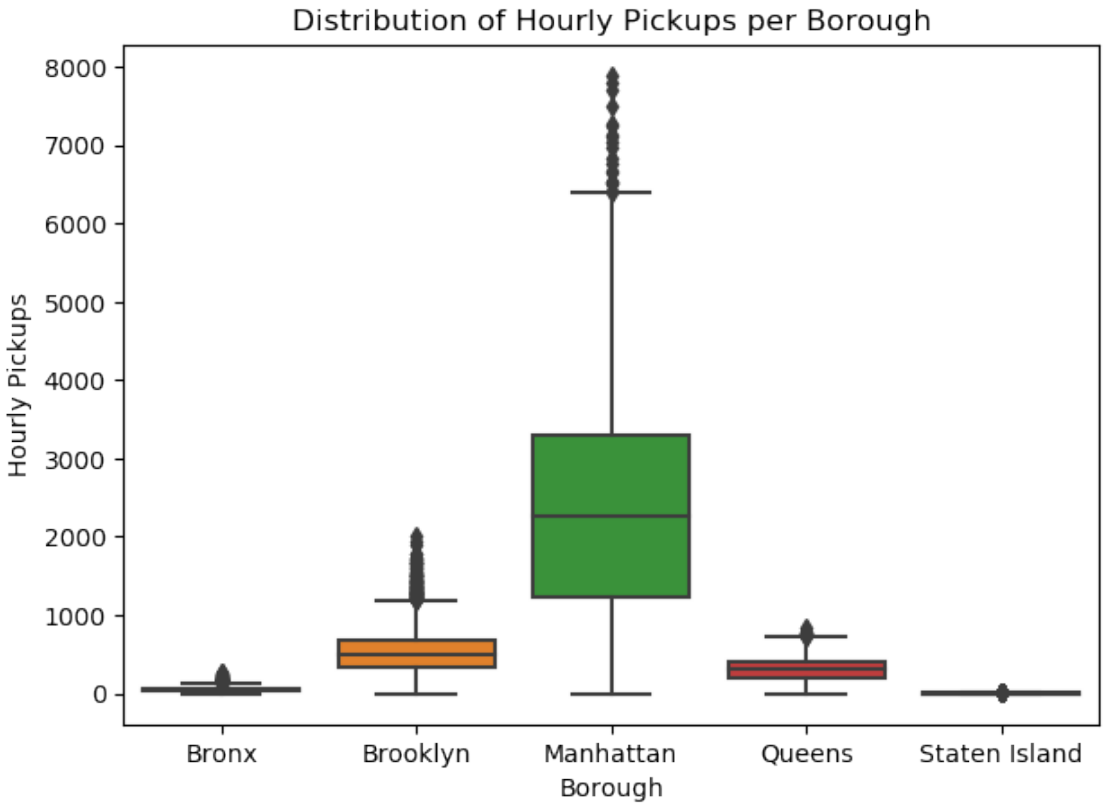
Before doing any analysis, it is important to investigate the data and clean it. As I looked through the data, I noticed that the maximum number of pickups for Newark Airport (EWR) was 2. Furthermore, there were multiple "NA" values for borough, indicating that the mapping of the pickup to the borough was unknown. The number of pickups for the "NA" borough was often in single digits as well, with a maximum of 11. Each of the five boroughs had 4343 entries, but the "NA" borough only had 3043. In order to have a consistent number of observations and to keep the analysis focused on just the five boroughs of New York City, I removed all the values corresponding to the "NA" and EWR boroughs. My exploratory data analysis was thus performed on the cleaned dataset.

[96]:
```
# Import libraries
import pandas as pd
import numpy as np

# Load data
uber = pd.read_csv("uber_nyc_enriched.csv")
len(uber[uber['borough'] == 'Manhattan']) # Each of the 5 boroughs has 4343␣
 ↪entries

# Rename "hday" column to "holiday"
uber = uber.rename(columns = {"hday":"holiday"})

# Check rows with "NaN" as a borough
uber_na = uber[uber['borough'].isnull()]
uber_na['pickups'].max() # 11
len(uber_na) # 3043 entries

# Check rows with "EWR" as a borough
uber_ewr = uber[uber['borough'] == 'EWR']
uber_ewr['pickups'].max() # 2
len(uber_ewr) # 4043 entries

# Remove EWR and NA boroughs from the analysis
uber_cleaned = uber.copy()
uber_cleaned = uber_cleaned[(uber_cleaned['borough'].notnull()) &␣
 ↪(uber_cleaned['borough'] != 'EWR')]
uber_cleaned;
```

6.1.3 Distribution of Pickups

I first looked at the distribution of hourly pickups across each borough, which is shown in the following boxplot. Manhattan clearly has the greatest average number of pickups and the largest distribution, followed by Brooklyn, Queens, Bronx, and Staten Island. Since Manhattan contains most of New York City's financial centers, cultural icons, and tourist attractions, it makes sense that it has the largest distribution of Uber pickups, even though it is not the most populous borough.

[184]:
```
# Histogram of pickup distribution across boroughs
import seaborn as sns
plt.figure(figsize = (7,5), dpi = 100);
pickup_dist = sns.boxplot('borough','pickups',data = uber_cleaned);
pickup_dist.set_title('Distribution of Hourly Pickups per Borough');
pickup_dist.set_xlabel('Borough');
pickup_dist.set_ylabel('Hourly Pickups');
```

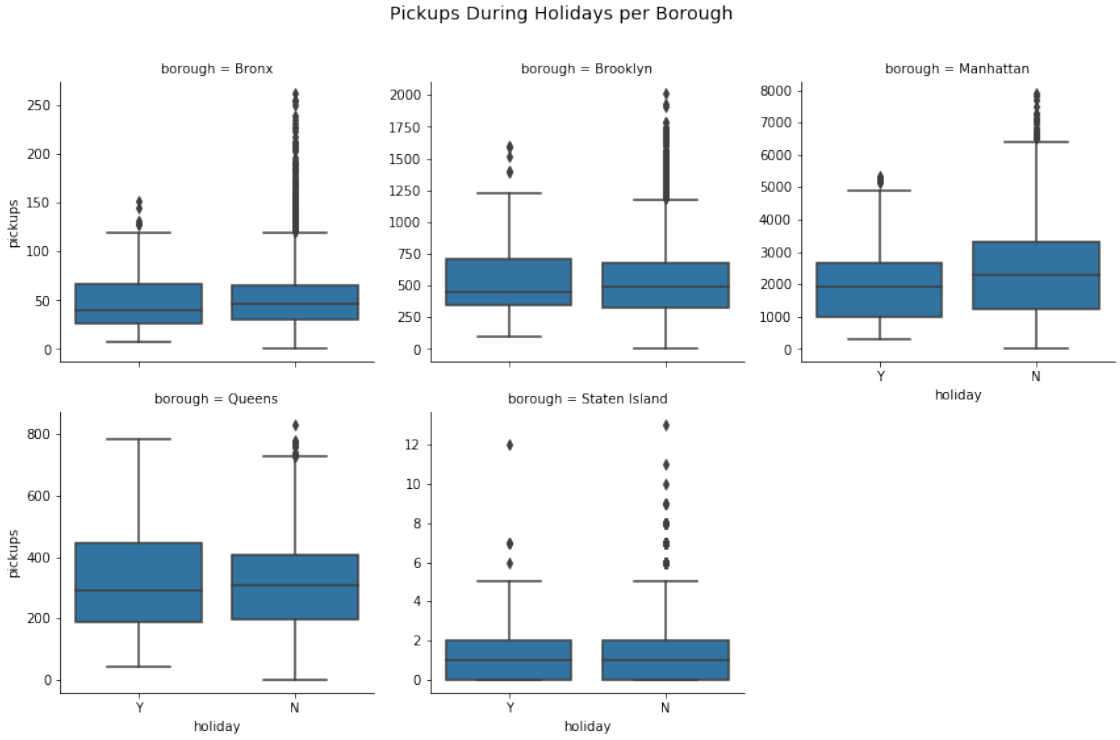


6.1.4 Investigating the effects of weather and holidays

Having observed the general distribution of pickups across each borough, I wanted to see if different factors were associated with noticeable changes in hourly pickups. The four variables of interest that I will show are the following: holiday, temperature, snow depth, and 24-hour precipitation.

Holidays Boxplots of hourly pickups for each borough during holidays and non-holidays are shown below. The dataset consisted of 7 holidays in the timeframe, the most notable of which are New Year's Day, Martin Luther King Jr. Day, Presidents' Day, and Memorial Day. From the plot, it is clear that the median number of pickups is not drastically different during holidays and non-holidays. In fact, they are very similar across boroughs. The distribution of pickups is larger during non-holidays, but this is likely because there are significantly more non-holidays than holidays, thus increasing the chances for variation.

[168]:
```
# Holidays
h_plot = sns.FacetGrid(col = 'borough', col_wrap = 3, data = uber_cleaned,␣
 ↪sharey = False, height = 4);
h_plot.map(sns.boxplot,'holiday','pickups');
h_plot.fig.suptitle('Pickups During Holidays per Borough', size = 14);
h_plot.fig.subplots_adjust(top=0.88)
plt.show()
```



Temperature Scatterplots of temperature vs. the number of hourly pickups per borough are shown below with trendlines. In all five boroughs, there is a general positive correlation between temperature and number of pickups. In warmer temperatures, people are more likely to go outside and travel, thus leading to more pickups.

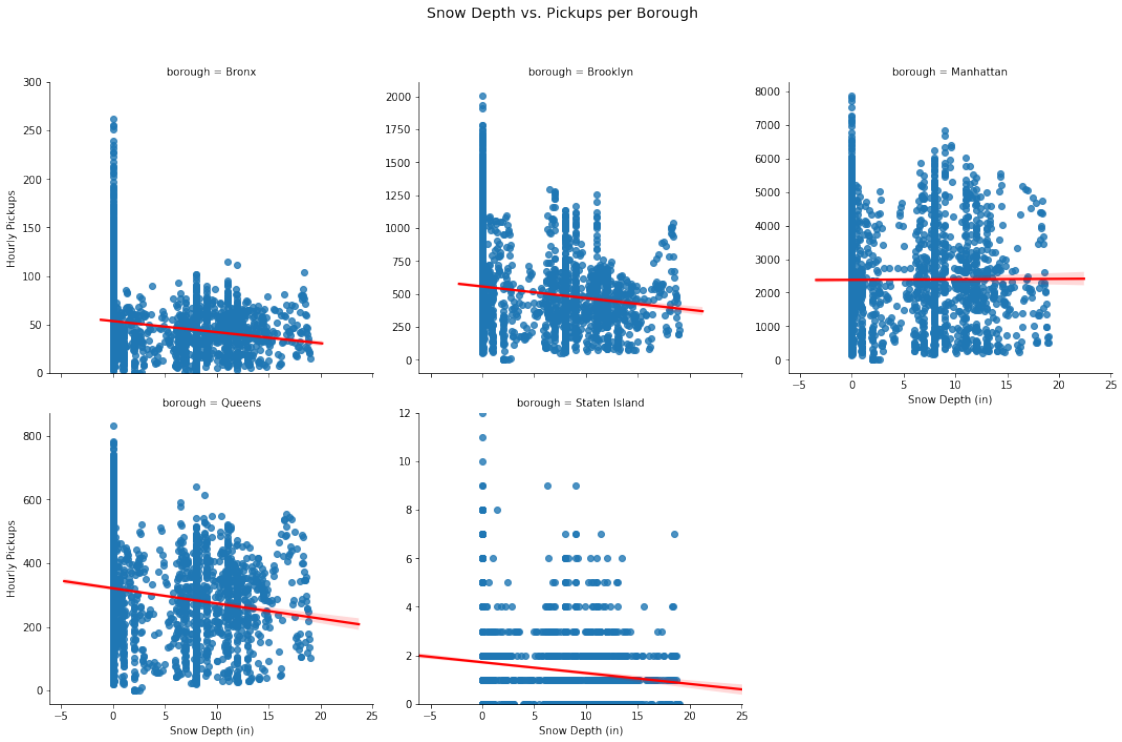
[186]:
```
```

```
# Temperature
t_plot = sns.lmplot(x = 'temp', y = 'pickups', col = 'borough', col_wrap = 3,␣
 ↪sharey = False,
                    data = uber_cleaned, line_kws = {'color': 'red'})
t_plot.axes[0].set_ylim(0,300)
t_plot.axes[4].set_ylim(0,12)
t_plot.set_axis_labels(x_var = 'Temperature (Fahrenheit)', y_var = 'Hourly␣
 ↪Pickups')
t_plot.fig.suptitle('Pickups During Holidays per Borough', size = 14);
t_plot.fig.subplots_adjust(top=0.88)
plt.show()
```

Snow Depth Scatterplots of snow depth vs. the number of hourly pickups per borough are shown below with trendlines. In all five boroughs, there is a general negative correlation between temperature and number of pickups, though it is least steep in Manhattan. These results provide further support to the above plots regarding temperature. Snow depth is more likely to increase as temperatures remain cold, so we should expect fewer pickups when there is more snow. In addition, poor road conditions due to high levels of snow will also likely reduce the number of people picked up by Uber.

[183]:
```
# Snow Depth
s_plot = sns.lmplot(x = 'sd', y = 'pickups', col = 'borough', col_wrap = 3,␣
 ↪sharey = False,
```

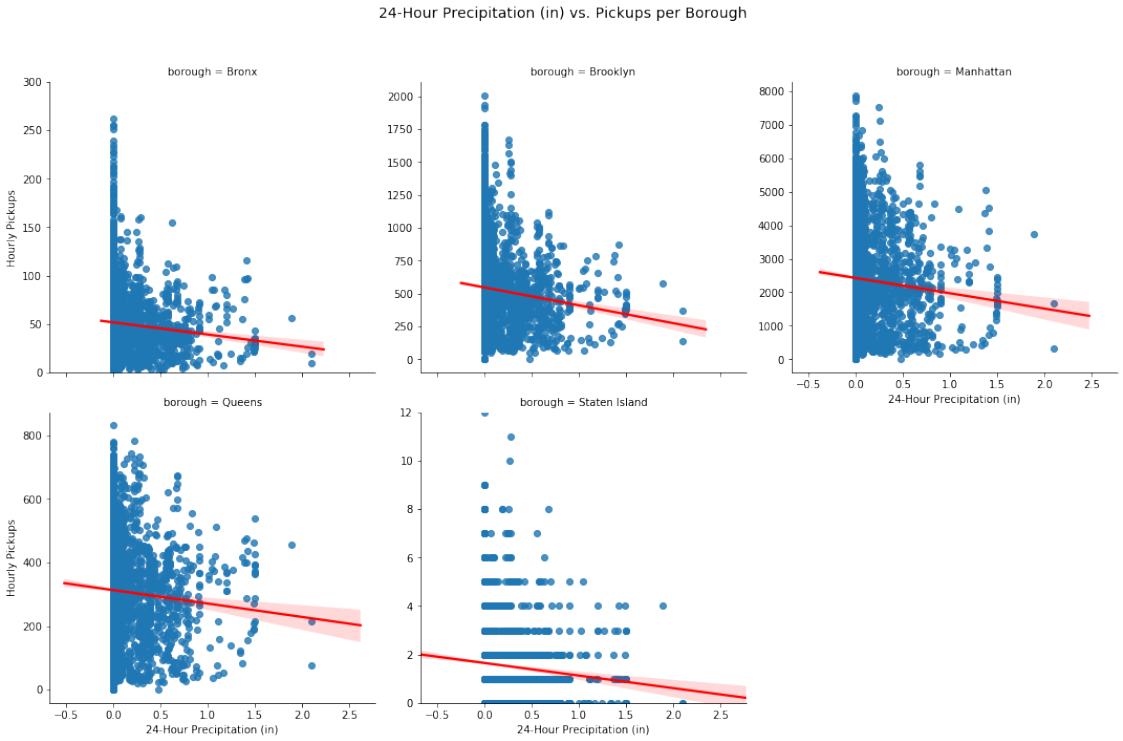
```
                    data = uber_cleaned, line_kws = {'color': 'red'})
s_plot.axes[0].set_ylim(0,300)
s_plot.axes[4].set_ylim(0,12)
s_plot.set_axis_labels(x_var = 'Snow Depth (in)', y_var = 'Hourly Pickups')
s_plot.fig.suptitle('Snow Depth vs. Pickups per Borough', size = 14);
s_plot.fig.subplots_adjust(top=0.88)
plt.show()
```



Precipitation Scatterplots of 24-hour precipitation vs. the number of hourly pickups per borough are shown below with trendlines. In all five boroughs, there is a general negative correlation between more rain and number of pickups. As the amount of rain increases, people are less likely to go outside and travel, thereby leading to fewer Uber pickups. Also, higher amounts of rain will make driving conditions more difficult, so fewer Ubers may be available, leading to fewer pickups.

[182]:
```
# 24-Hour Precipitation
p24_plot = sns.lmplot(x = 'pcp24', y = 'pickups', col = 'borough', col_wrap =␣
 ↪3, sharey = False,
                    data = uber_cleaned, line_kws = {'color': 'red'})
p24_plot.axes[0].set_ylim(0,300)
p24_plot.axes[4].set_ylim(0,12)
p24_plot.set_axis_labels(x_var = '24-Hour Precipitation (in)', y_var = 'Hourly␣
 ↪Pickups')
```

```
p24_plot.fig.suptitle('24-Hour Precipitation (in) vs. Pickups per Borough',␣
 ↪size = 14);
p24_plot.fig.subplots_adjust(top=0.88)
plt.show()
```



6.1.5 Conclusions

Overall, this basic exploratory data analysis highlighted some of the key factors that influence the number of hourly Uber pickups across the boroughs of New York City. In particular, the plots show that holidays do not appear to significantly affect the number of pickups, higher temperatures generally lead to more pickups, and higher levels of snow and rain generally lead to fewer pickups. To expand on this further, we can plot the other variables and make a statistical model using all of these factors to predict the number of Uber pickups across each borough. A model could also be built using several years of Uber and weather data, instead of just 6 months' worth of data, to more accurately assess which variables are significant in influencing Uber pickups. Finally, a time series model of Uber pickups and weather data could be developed, allowing a company such as Uber to predict their number of pickups based on forecasted weather data.