Assignment 1 - Probability, Linear Algebra, & Computational Programming

Yuanjing Zhu

Netid: yz792

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 is also linked to from the course syllabus.

Total points in the assignment add up to 90; an additional 10 points are allocated to 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 you 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. For this assignment show ALL math work for questions 1-4, meaning that you should include any intermediate steps necessary to understand the logic of your solution

[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

Properties of a probability density function describe the rules that a probability density function needs to follow:

> 1. The function needs to be greater than zero
> 2. The total area under the curve of the function is equal to one
> 3. The function can be any real positive number

Therefore:

$0 + \int_0^2 \alpha x^2 dx + 0 = 1$

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

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

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

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

The CDF, F(x), is area function of the PDF, obtained by integrating the PDF from negative infinity to an arbitrary value x.

If x is in the interval $(-\infty, 0]$, then:

$$\begin{aligned} F(x) &= \int_{-\infty}^{0} f(t)dt \\ &= \int_{-\infty}^{0} 0dt \\ &= 0 \end{aligned}$$

If x is in the interval $(0, 3)$, then:

$$\begin{aligned} F(x) &= \int_{-\infty}^{x} f(t)dt \\ &= \int_{-\infty}^{0} f(t)dt + \int_{0}^{x} f(t)dt \\ &= 0 + \int_{0}^{x} \frac{1}{3}dt \\ &= \frac{1}{3}t\Big|_0^x \\ &= \frac{1}{3}x \end{aligned}$$

If x is in the interval $[3, \infty)$ then:

$$\begin{aligned} F(x) &= \int_{-\infty}^{x} f(t)dt \\ &= \int_{-\infty}^{0} f(t)dt + \int_{0}^{3} f(t)dt + \int_{3}^{x} f(t)dt \\ &= 0 + \int_{0}^{3} \frac{1}{3}dt + 0 \\ &= \frac{1}{3}t\Big|_0^3 \\ &= 1 \end{aligned}$$

The CDF is therefore given by:

$$F(x) = \begin{cases} 0 & x \leqslant 0 \\ \frac{1}{3}x & 0 < x < 3 \\ 1 & x \geqslant 3 \end{cases}$$

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

a) Expected value of X:

$$\begin{aligned} E(X) &= \int_{-\infty}^{\infty} xf(x)dx \\ &= 0 + \int_{0}^{3} \frac{1}{3}xdx + 0 \\ &= \frac{1}{6}x^2|_0^3 \\ &= \frac{3}{2} \end{aligned}$$

b) Variance of X

$Var(X) = E(X^2) - (E(X))^2$

$$\begin{aligned} E(X^2) &= \int_{-\infty}^{\infty} x^2 f(x)dx \\ &= 0 + \int_{0}^{3} \frac{1}{3}x^2dx + 0 \\ &= \frac{1}{9}x^3|_0^3 \\ &= 3 \end{aligned}$$

$E(X) = \frac{3}{2}$

Therefore: $Var(X) = 3 - (\frac{3}{2})^2 = \frac{3}{4}$

4

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

Table 1. Dataset N=5 observations

	x_0	x_1	x_2	x_3	x_4
$\mathbf{x}$	2	3	10	-1	-1

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

Show all work. Your answer should include the definition of mean and variance in the context of discrete data. In this case, use the sample variance since the sample size is quite small

ANSWER

a) The mean of discrete data is a measure of central tendency that gives an idea of "average" value of a dataset. It is defined as the sum of all the observations divided by the total number of observations in the dataset.

$$\begin{aligned} E(X) &= \frac{\sum_0^N x_i}{N} \\ &= \frac{2+3+10-1-1}{5} \\ &= 2.6 \end{aligned}$$

b) The variance of discrete data is a measure of the spread or dispersion of the data around the mean. It is defined as the average of the squared deviations of each data point from the mean. Since the sample size is quite small, we use sample variance which uses (n-1) instead of n as the denominator to correct the bias in the estimation of the population variance

$$\begin{aligned} Var(X) &= \frac{\sum_0^N (x_i-\mu)^2}{N-1} \\ &= \frac{(2-2.6)^2+(3-2.6)^2+(10-2.6)^2+(-1-2.6)^2+(-1-2.6)^2}{5-1} \\ &= 20.3 \end{aligned}$$

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) Suppose the matrix after dimensionality reduction is denoted as $\mathbf{B}$. Projecting the sample of data $\mathbf{x}$ on the new basis defined by $\mathbf{A}$ can be represented as: $\mathbf{Ax} = \mathbf{B}$. The dimension of $\mathbf{x}$ is (4, 1) and the dimension of $\mathbf{B}$ is (2, 1), so according to matrix multiplication the dimension of $\mathbf{A}$ is (2, 4) where each column of the matrix $\mathbf{A}$ represents a new basis vector in the transformed space, and each row represents the coefficients of the original basis vectors in the new space.

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

$$\mathbf{Ax} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 \end{bmatrix}$$

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.

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 **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
```

```
In [ ]: # create matrix
        A = np.array([[1, 2, 3], [2, 4, 5], [3, 5, 6]])
        b = np.array([-1, 3, 8]).reshape(3, 1)
        c = np.array([4, -3, 6]).reshape(3, 1)
        I = np.eye(3)
```

1. $\mathbf{A}\mathbf{A}$

```
In [ ]: np.dot(A, A)

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

2. $\mathbf{A}\mathbf{A}^T$

```
In [ ]: np.dot(A, A.T)

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

3. $\mathbf{A}\mathbf{b}$

```
In [ ]: np.dot(A, b)

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

4. $\mathbf{A}\mathbf{b}^T$

The operation is invalid because the number of columns in the first matrix is not equal to the number of rows in the second matrix since the dimension of $\mathbf{A}$ is $(3, 3)$ but the dimension of $\mathbf{b^T}$ is $(1, 3)$.

5. $\mathbf{b}\mathbf{A}$

The operation is invalid because the number of columns in the first matrix is not equal to the number of rows in the second matrix since the dimension of $\mathbf{b}$ is $(3, 1)$ but the dimension of $\mathbf{A}$ is $(3, 3)$.

6. $\mathbf{b}^T\mathbf{A}$

```
In [ ]: np.dot(b.T, A)

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

7. $\mathbf{b}\mathbf{b}$

The operation is invalid because the number of columns in the first matrix is not equal to the number of rows in the second matrix since the dimension of $\mathbf{b}$ is $(3, 1)$.

8. $\mathbf{b}^T\mathbf{b}$

```
In [ ]: np.dot(b.T, b)

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

9. $\mathbf{b}\mathbf{b}^T$

```
In [ ]: np.dot(b, b.T)
```

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

10. $\mathbf{b} + \mathbf{c}^T$

The operation is invalid referring to underlying linear algebra because two matrices cannot be added if their dimensions are not the same. The dimension of $\mathbf{b}$ is $(3, 1)$, but the dimension of $\mathbf{c^T}$ is $(1, 3)$.

11. $\mathbf{b}^T\mathbf{b}^T$

The operation is invalid because the number of columns in the first matrix is not equal to the number of rows in the second matrix since the dimension of $\mathbf{b^T}$ is $(1, 3)$.

12. $\mathbf{A}^{-1}\mathbf{b}$

```
In [ ]: np.dot(np.linalg.inv(A), b)
```

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

13. $\mathbf{A} \circ \mathbf{A}$

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

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

14. $\mathbf{b} \circ \mathbf{c}$

```
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 λ, 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

1. Calculate the eigenvalues and corresponding eigenvectors of matrix $\mathbf{A}$

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

# calculate the eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(A)
for i in range(3):
    print('Eigenvalue: ', eigenvalues[i])
    print('Corresponding eigenvector: ', eigenvectors[:, i])
    print("======================================")
```

```
Eigenvalue:  11.344814282762078
Corresponding eigenvector:  [-0.32798528 -0.59100905 -0.73697623]
======================================
Eigenvalue:  -0.5157294715892571
Corresponding eigenvector:  [-0.73697623 -0.32798528  0.59100905]
======================================
Eigenvalue:  0.17091518882717882
Corresponding eigenvector:  [ 0.59100905 -0.73697623  0.32798528]
======================================
```

2. Here I choose the first pair of eigenvalue and eigenvector to validate $\mathbf{Av} = \lambda\mathbf{v}$ and $\mathbf{AAv} = \lambda^2\mathbf{v}$

In [ ]:
```python
e_value1 = eigenvalues[0]
e_vect1 = eigenvectors[:, 0]

print("Av - λv = ", np.dot(A, e_vect1) - e_value1 * e_vect1)
print("It's extremely close to zero, so Av = λv")
print("AAv - λ^2v = ", np.dot(A, np.dot(A, e_vect1)) - (e_value1 **2) * e_vect1)
print("It's extremely close to zero, so AAv = λ^2v")
```

```
Av - λv =  [-1.77635684e-15 -2.66453526e-15  5.32907052e-15]
It's extremely close to zero, so Av = λv
AAv - λ^2v =  [-2.13162821e-14 -1.42108547e-14  4.26325641e-14]
It's extremely close to zero, so AAv = λ^2v
```

In [ ]:
```python
e_vect1 = eigenvectors[:, 0]
e_vect2 = eigenvectors[:, 1]
e_vect3 = eigenvectors[:, 2]

print("The inner product of eigenvector1 and eigenvector2 is: ", np.inner(e_vect1, e_vect2))
print("It's extremely close to zero, so eigenvector1 and eigenvector2 are orthogonal")
print("======================================")
print("The inner product of eigenvector1 and eigenvector3 is: ", np.inner(e_vect1, e_vect3))
print("It's extremely close to zero, so eigenvector1 and eigenvector3 are orthogonal")
print("======================================")
print("The inner product of eigenvector2 and eigenvector3 is: ", np.inner(e_vect2, e_vect3))
print("It's extremely close to zero, so eigenvector2 and eigenvector3 are orthogonal")
```

```
The inner product of eigenvector1 and eigenvector2 is:  -3.608224830031759e-16
It's extremely close to zero, so eigenvector1 and eigenvector2 are orthogonal
======================================
The inner product of eigenvector1 and eigenvector3 is:  -1.1102230246251565e-16
It's extremely close to zero, so eigenvector1 and eigenvector3 are orthogonal
======================================
The inner product of eigenvector2 and eigenvector3 is:  -1.3600232051658168e-15
It's extremely close to zero, so eigenvector2 and eigenvector3 are 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

(a) The plant that has generated the most energy (MWh) is:

In [ ]:
```python
import numpy as np
import pandas as pd

# read the data
a1_egrid = pd.read_excel("F:/Duke MIDS/705_ML/Assignment/01/a1_egrid2016.xlsx", header=1, engine='openpyxl')
a1_egrid.head()
```

Out[ ]:

	SEQPLT16	PSTATABB	PNAME	LAT	LON	PLPRMFL	CAPFAC	NAMEPCAP	PLNGENAN	PLCO2EQA
0	1	AK	7-Mile Ridge Wind Project	63.210689	-143.247156	WND	NaN	1.8	NaN	NaN
1	2	AK	Agrium Kenai Nitrogen Operations	60.673200	-151.378400	NG	NaN	21.6	NaN	NaN
2	3	AK	Alakanuk	62.683300	-164.654400	DFO	0.05326	2.6	1213.001	1049.863
3	4	AK	Allison Creek Hydro	61.084444	-146.353333	WAT	0.01547	6.5	881.000	0.000
4	5	AK	Ambler	67.087980	-157.856719	DFO	0.13657	1.1	1315.999	1087.881

In [ ]:
```
max_PLNGENAN = a1_egrid["PLNGENAN"].max()
max_PLNGENAN_PNAME = a1_egrid.loc[a1_egrid["PLNGENAN"] == max_PLNGENAN, "PNAME"].values[0]
print(f"The plant that has generated the most engergy is {max_PLNGENAN_PNAME}.")
```

```
The plant that has generated the most engergy is Palo Verde.
```

(b) The northern-most power plant in the United States is:

In [ ]:
```
max_LAT = a1_egrid["LAT"].max()
max_LAT_PNAME = a1_egrid.loc[a1_egrid["LAT"] == max_LAT, "PNAME"].values[0]
print(f"The northern-most power plant in the US is {max_LAT_PNAME}.")
```

```
The northern-most power plant in the US is Barrow.
```

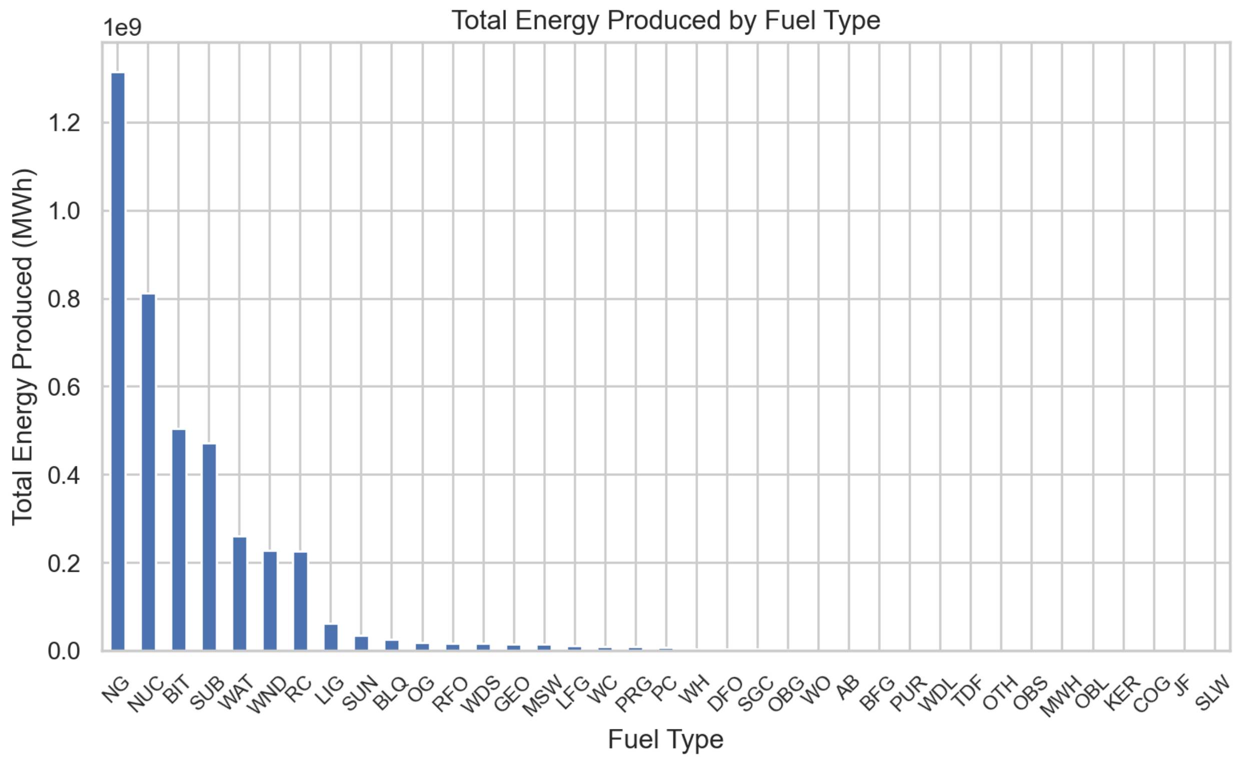
(c) The state where the northern-most power plant in the United States is located?

In [ ]:
```
max_LAT_PSTATABB = a1_egrid.loc[a1_egrid["LAT"] == max_LAT, "PSTATABB"].values[0]
print(f"The state of the northern-most power plant in the US is {max_LAT_PSTATABB}.")
```

```
The state of the northern-most power plant in the US is AK.
```

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

In [ ]:
```
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

plt.figure(figsize=(8,5), dpi= 100)
a1_egrid.groupby("PLPRMFL")["PLNGENAN"].sum().sort_values(ascending=False).plot(kind="bar")
plt.xlabel("Fuel Type")
plt.ylabel("Total Energy Produced (MWh)")
plt.xticks(rotation=45, fontsize=9)
plt.title("Total Energy Produced by Fuel Type")
plt.tight_layout()
plt.show()
```



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

From the plot in (d), NG, which stands for natural gas, produces the most engergy (MWh) in the US.

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. 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 [ ]:
```
# create an array of 10 million random numbers
np.random.seed(123)     # set random seed
x = np.random.randn(10000000)
```

```
In [ ]: import time
        start = time.time()
        ss = 0
        for i in range(x.shape[0]):
            ss += x[i] ** 2
        end = time.time()
        non_vectorized_time = end - start
        print(f"Non-vectorized time: {non_vectorized_time:.4f} seconds")
```

```
Non-vectorized time: 3.3446 seconds
```

```
In [ ]: start = time.time()
        np.dot(x, x)
        end = time.time()
        vectorized_time = end - start
        print(f"Vectorized time: {vectorized_time:.4f} seconds")
```

```
Vectorized time: 0.0050 seconds
```

```
In [ ]: print(f"TIME [sec] (not-vectorized): {non_vectorized_time:.4f}")
        print(f"TIME [sec] (vectorized): {vectorized_time:.4f}")
        print(f"The vectorized code is {(non_vectorized_time / vectorized_time):.0f} \
                times faster than the nonvectorized code")
```

```
TIME [sec] (not-vectorized): 3.3446
TIME [sec] (vectorized): 0.0050
The vectorized code is 665 times faster than the nonvectorized code
```

10

[10 points] This exercise will walk through some basic numerical programming and probabilistic thinking exercises, two skills which are frequently use 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) Generate random variable X

```
In [ ]: # generate 10^4 normally distributed random numbers with mean of 2 and standard deviation of 1
        np.random.seed(123)       # set random seed
        x = np.random.normal(2, 1, 10000)
```

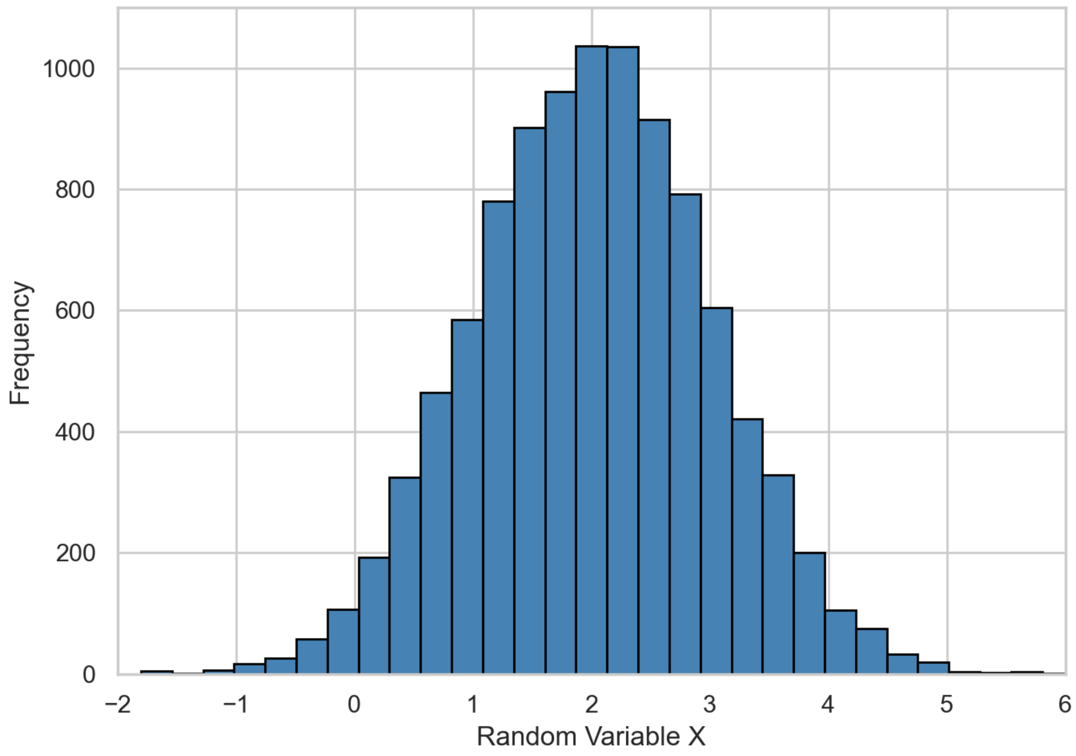
2) Calculate the mean and standard deviation of x

```
In [ ]: print(f"The mean of x is: {np.mean(x):.4f}")
        print(f"The standard deviation of x is: {np.std(x):.4f}")
```

```
The mean of x is: 2.0097
The standard deviation of x is: 0.9981
```

3) Plot the histogram of the data in x

In [ ]:
```
import matplotlib.pyplot as plt
%config InlineBackend.figure_format = 'retina'

# Create the plot
plt.figure(figsize=(7,5), dpi= 100)
plt.hist(x, bins=30, color='steelblue', edgecolor='black', linewidth=1.0,)
plt.xlabel('Random Variable X')
plt.ylabel('Frequency')
plt.axis([-2, 6, 0, 1100])
plt.tight_layout() # Use this to maximize the use of space in the figure
plt.show()
```



4) Calculate the 90th percentile of x

The formula to compute percentiles of a normal distribution is $X = \mu + Z\sigma$. After looking at the Z values table for commonly used percentiles, $Z = 1.282$ when it's 90th percentile. So $X = 2 + 1.282 * 1 = 3.282$

In [ ]:
```
# using numpy
p90 = np.percentile(x, 90)
print(f"The 90th percentile of x is: {p90:.3f}")
```

```
The 90th percentile of x is: 3.289
```

5) Generate the 99th percentile of x

$Z = 2.326$ when it's 99th percentile. So $X = \mu + Z\sigma = 2 + 2.326 * 1 = 4.326$

In [ ]:
```
# using numpy
p99 = np.percentile(x, 99)
print(f"The 99th percentile of x is: {p99:.3f}")
```

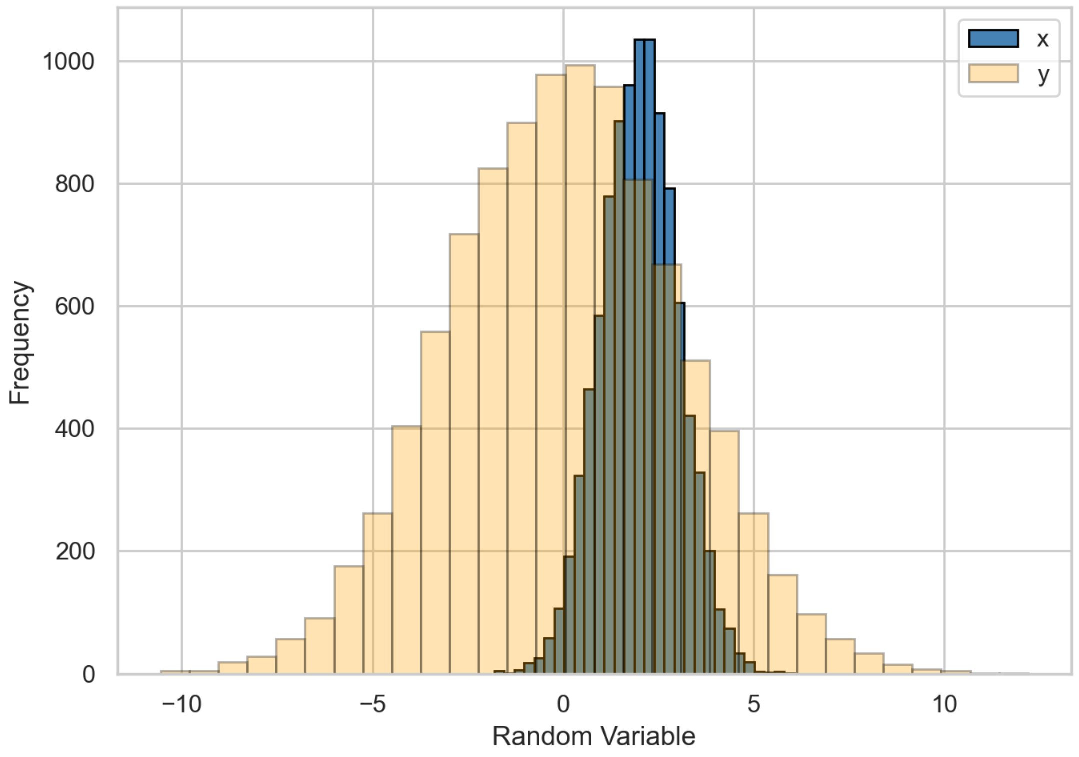
```
The 99th percentile of x is: 4.326
```

6) Generate random variable Y

In [ ]:
```
y = np.random.normal(0, 3, 10000)
```

7) Create a new figure and plot the histogram of the data in $\mathbf{y}$ on the same axes with the histogram of $\mathbf{x}$

In [ ]:
```
plt.figure(figsize=(7,5), dpi= 100)
plt.hist(x, bins=30, color = "steelblue", edgecolor='black', linewidth=1.0, alpha=1.0)
plt.hist(y, bins=30, color = "orange", edgecolor='black', linewidth=1.0, alpha=0.3)
plt.legend(['x', 'y'])
plt.xlabel('Random Variable')
plt.ylabel('Frequency')
plt.tight_layout() # Use this to maximize the use of space in the figure
plt.show()
```



8) Using the observations from $\mathbf{x}$ and $\mathbf{y}$, estimate $E[XY]$

X and Y are independent, so E(XY) = E(X)E(Y) = 2 * 0 = 0

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.

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, [Yuanjing Zhu], affirm that I have [completed the above tutorial / I 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 a ~3 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 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

(a) Motivation

As the 5G era dawns, the significance of telecom networks as infrastructure has become increasingly crucial, and competition in the telecom market has grown increasingly fierce. In this context, utilizing customer information and behavior to inform more precise marketing, predicting customer loss, and identifying key factors that influence customer churn can assist product teams in adjusting their marketing strategies and developing retention measures.

I found a dataset from Kaggle, which is shared on the IBM Business Analytics Community in 2019 Q3. The dataset comprises records from a telecom company providing telephone and internet services to over 7,000 users (individuals and households) in California. It includes customers' account information, demographic information, registered services, and a "churn" label. I consider this dataset to be suitable for analyzing customer churn because:

1. It has been sourced from a reputable source (IBM), ensuring reliability, accuracy, completeness, and high quality.
2. It comprises a range of important information, including customers' service, account, and demographic data, which is essential in building an accurate profile of customer behavior.
3. With over 70,000 records, it is of a sufficient size to provide robust and representative statistical power for the analysis.

The dataset contains 7043 rows (customers) and 21 columns (features). The "Churn" column is our target. Each feature is described below.

Feature	Description
customerID	A unique ID that identifies each customer.
gender	The customer's gender: Male, Female
SeniorCitizen	Indicates if the customer is 65 or older: Yes, No
Partner	Indicates if the customer has a partner or not: Yes, No
Dependents	Indicates if the customer lives with any dependents: Yes, No. Dependents could be children, parents, grandparents, etc.
tenure	Number of months the customer has stayed with the company
PhoneService	Indicate if the customer has a phone service or not: Yes, No
MultipleLines	Indicate if the customer has multiple lines or not: Yes, No, No phone service
InternetService	Customer's internet service provider: DSL, Fiber optic, No
OnlineSecurity	Indicate if the customer has online security or not: Yes, No, No internet service
OnlineBackup	Indicates if the customer subscribes to an additional online backup service provided by the company: Yes, No, No internetservice
DeviceProtection	Indicates if the customer subscribes to an additional device protection plan for their Internet equipment provided by the company: Yes, No, No internetservice
TechSupport	Indicates if the customer subscribes to technical support plan: Yes, No, No internetservice
StreamingTV	Indicates if the customer uses their Internet service to stream television programing from a third party provider: Yes, No, No internetservice
StreamingMovies	Indicates if the customer uses their Internet service to stream movies from a third party provider: Yes, No, No internetservice
Contract	Indicates the customer's current contract type: Month-to-Month, One Year, Two Year.
PaperlessBilling	Indicates if the customer has chosen paperless billing: Yes, No
PaymentMethod	Indicates how the customer pays their bill: bank transfer, credit card, electronic check, mailed check
MonthlyCharges	Indicates the customer's current total monthly charge for all their services from the company.
TotalCharges	Indicates the customer's total charges, calculated to the end of the quarter specified above.

After conducting thorough data cleaning and visualization, I will identify the customer characteristics that are more likely to result in churn and those that are more likely to result in retention. I will use this information to provide guidance on marketing strategies to help the company reduce the risk of churn.

(b) Data cleaning

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
%config InlineBackend.figure_format = 'retina'
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
tcc = pd.read_csv("F:/Duke MIDS/705_ML/Assignment/01/WA_Fn-UseC_-Telco-Customer-Churn.csv")
tcc.head()
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...

5 rows × 21 columns

In [ ]:
```
tcc.shape
```

Out[ ]:
```
(7043, 21)
```

The dataset has 7043 rows and 21 columns.

In [ ]:
```
tcc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

There are 3 numeric features: SeniorCitizen, tenure, MonthlyCharges. Others are categorical variables.

Check for duplicates

```
In [ ]: print(f"There are {tcc.duplicated().sum()} duplicated rows.")
        print(f"Whether number of customer ID is equal to number of rows: \
              {tcc['customerID'].nunique() == tcc.shape[0]}")
```

```
There are 0 duplicated rows.
Whether number of customer ID is equal to number of rows:       True
```

There are no duplicated customer ID nor duplicated rows.

Check for missing values

```
In [ ]: tcc.isnull().sum()
```

```
Out[ ]: customerID          0
        gender              0
        SeniorCitizen       0
        Partner             0
        Dependents          0
        tenure              0
        PhoneService        0
        MultipleLines       0
        InternetService     0
        OnlineSecurity      0
        OnlineBackup        0
        DeviceProtection    0
        TechSupport         0
        StreamingTV         0
        StreamingMovies     0
        Contract            0
        PaperlessBilling    0
        PaymentMethod       0
        MonthlyCharges      0
        TotalCharges        0
        Churn               0
        dtype: int64
```

There are no NaNs in all features.

Notice that the feature "TotalCharges" are numbers, indicating it should be numeric variable. But executing "tcc['TotalCharges'].astype(float)" raises error. Therefore, there might be string in "TotalCharges" column.

```
In [ ]: # check string in TotalCharges
        tcc['TotalCharges'].value_counts().sort_values(ascending=False)
```

```
Out[ ]:             11
        20.2        11
        19.75        9
        20.05        8
        19.9         8
                    ..
        3886.45      1
        1224.05      1
        2310.2       1
        723.4        1
        6844.5       1
        Name: TotalCharges, Length: 6531, dtype: int64
```

Here space " " is the missing value in "TotalCharges" .

```
In [ ]: # replace the space with NaN
        tcc['TotalCharges'] = tcc['TotalCharges'].replace(" ", np.nan)

        # convert the column to float
        tcc['TotalCharges'] = tcc['TotalCharges'].astype(float)
```

```
In [ ]: # check the missing values in the column
        print(f"There are {tcc['TotalCharges'].isnull().sum()} missing values in the column.")
```

```
# check the missing values
tcc[tcc['TotalCharges'].isnull()]
```

```
There are 11 missing values in the column.
```

Out[ ]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
488	4472-LVYGI	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes
753	3115-CZMZD	Male	0	No	Yes	0	Yes	No	No	No internet service
936	5709-LVOEQ	Female	0	Yes	Yes	0	Yes	No	DSL	Yes
1082	4367-NUYAO	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service
1340	1371-DWPAZ	Female	0	Yes	Yes	0	No	No phone service	DSL	Yes
3331	7644-OMVMY	Male	0	Yes	Yes	0	Yes	No	No	No internet service
3826	3213-VVOLG	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service
4380	2520-SGTTA	Female	0	Yes	Yes	0	Yes	No	No	No internet service
5218	2923-ARZLG	Male	0	Yes	Yes	0	Yes	No	No	No internet service
6670	4075-WKNIU	Female	0	Yes	Yes	0	Yes	Yes	DSL	No
6754	2775-SEFEE	Male	0	No	Yes	0	Yes	Yes	DSL	Yes

11 rows × 21 columns

The tenure of these 11 users are all recorded as 0, indicating that they joined the network just before the end of the data collection period, and therefore have no previous records. Despite having a payment record for the current month, we can assume that their previous payment records are 0 and record these missing values as such.

In [ ]:
```
# replace missing values with 0
tcc['TotalCharges'] = tcc['TotalCharges'].fillna(0)

# make sure there is no missing value
tcc['TotalCharges'].isnull().sum()
```

Out[ ]:
```
0
```

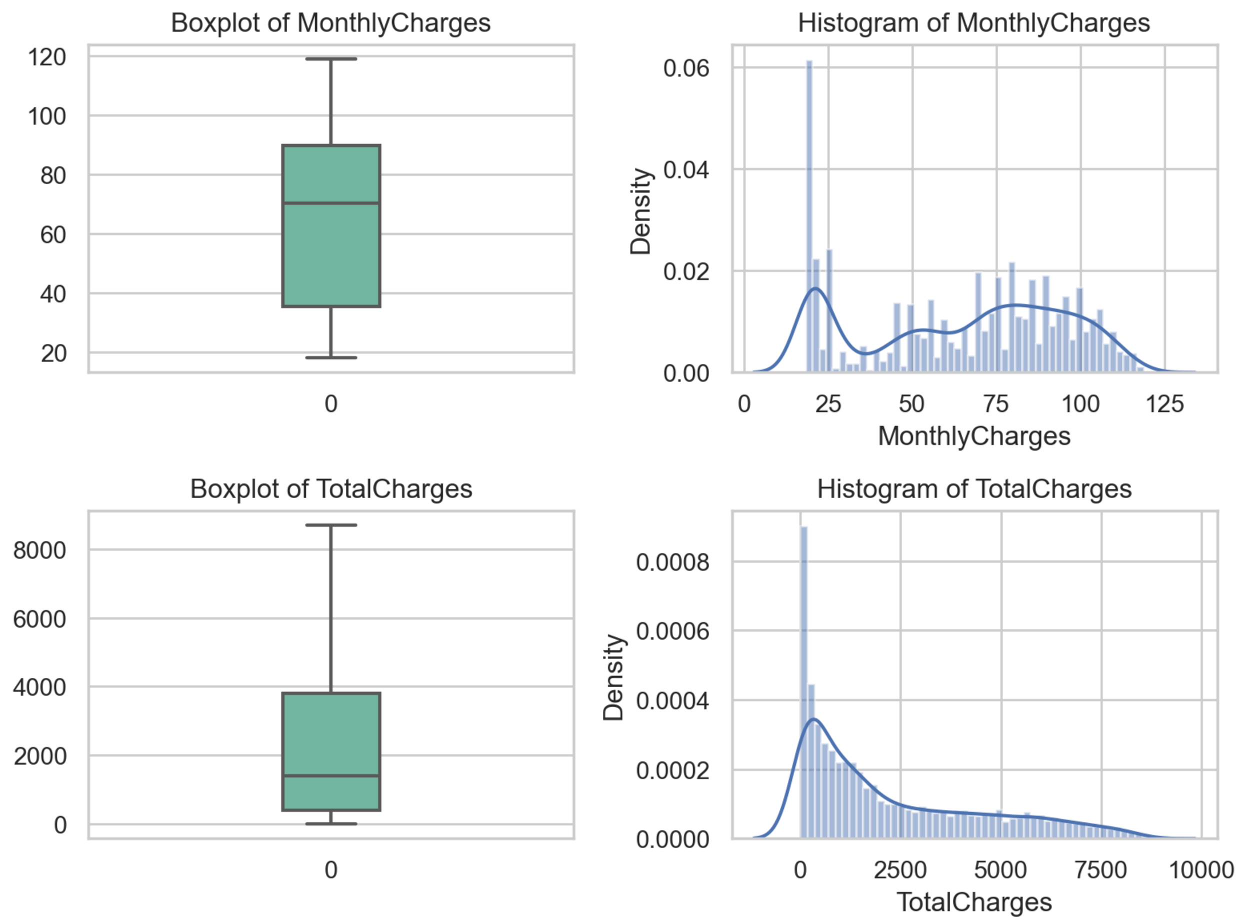
Check for outliers

In [ ]:
```
tcc.describe()
```

Out[ ]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

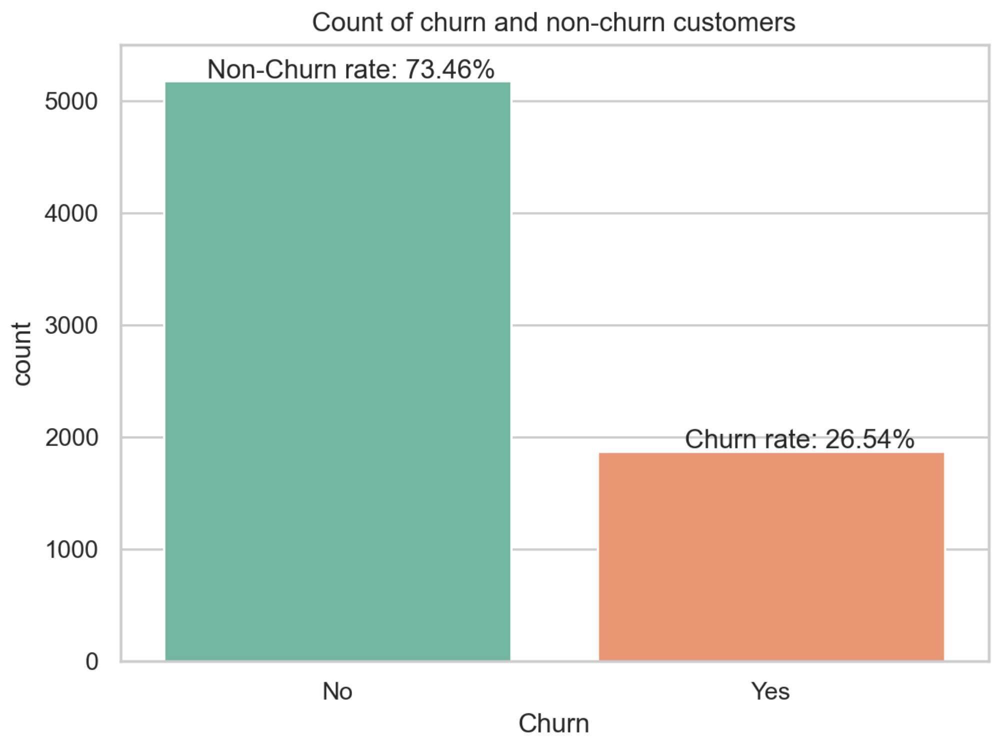
In [ ]:
```python
fig, axes = plt.subplots(ncols = 2, nrows = 2, figsize = (8, 6))

sns.boxplot(data = tcc['MonthlyCharges'], width = 0.2, ax = axes[0, 0], palette = 'Set2')
axes[0, 0].set(title = 'Boxplot of MonthlyCharges')

sns.distplot(tcc['MonthlyCharges'], ax = axes[0, 1], kde = True, bins = 50, hist_kws={'alpha': 0.5})
axes[0, 1].set(title = 'Histogram of MonthlyCharges')

sns.boxplot(data = tcc['TotalCharges'], width = 0.2, ax = axes[1, 0], palette = 'Set2')
axes[1, 0].set(title = 'Boxplot of TotalCharges')

sns.distplot(tcc['TotalCharges'], ax = axes[1, 1], kde = True, bins = 50, hist_kws={'alpha': 0.5})
axes[1, 1].set(title = 'Histogram of TotalCharges')

plt.tight_layout()
plt.show()
```



There are no outliers.

Distribution of label

In [ ]:
```python
churn_percentage = round((len(tcc[tcc['Churn'] == 'Yes']) / tcc.shape[0]) * 100, 2)
non_churn_percentage = round((len(tcc[tcc['Churn'] == 'No']) / tcc.shape[0]) * 100, 2)
plt.figure(figsize = (7, 5))
sns.countplot(x = 'Churn', data = tcc, dodge = False, palette = 'Set2')
plt.title('Count of churn and non-churn customers')
plt.ylim([0, 5500])
plt.text(0.8, 1900, s = 'Churn rate: ' + str(churn_percentage) + '%' )
plt.text(-0.3, 5200, s = 'Non-Churn rate: ' + str(non_churn_percentage) + '%' )
plt.show()
```



Customer churn rate is 26.54% over 7043 records, which is pretty high.

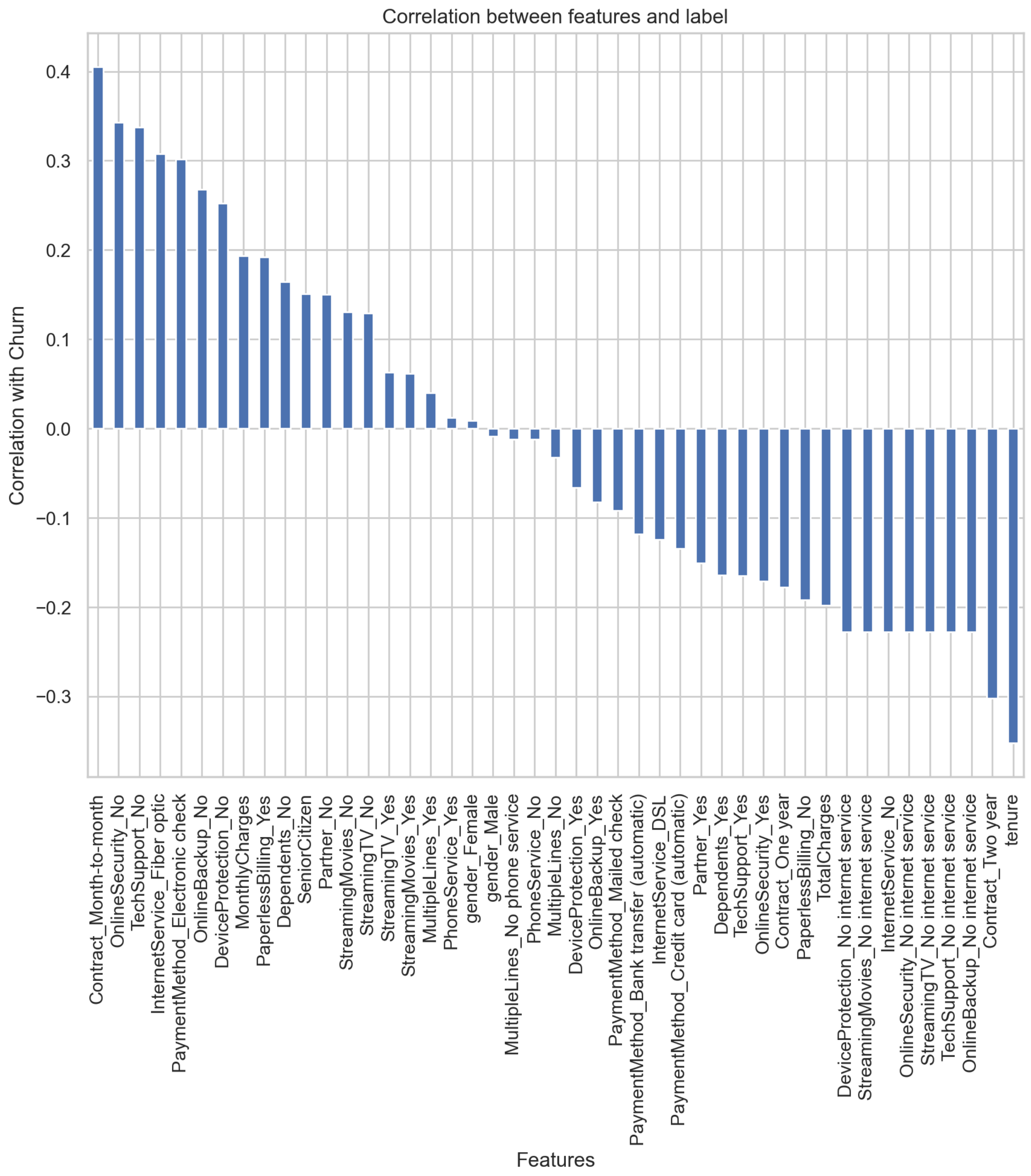
(c) Data visualization

Correlation

```
In [ ]: # drop column id
        tcc_tmp = tcc.iloc[:, 1:].copy()

        # convert Yes/No in label to 1/0
        tcc_tmp['Churn'].replace('Yes', 1, inplace = True)
        tcc_tmp['Churn'].replace('No', 0, inplace = True)

        # Convert to dummy variables
        tcc_dummy = pd.get_dummies(tcc_tmp)

        # correlation between features and label
        tcc_dummy_corr = tcc_dummy.corr()
        tcc_dummy_corr['Churn'].sort_values(ascending = False)
```

Out[ ]:
```
Churn                                      1.000000
Contract_Month-to-month                    0.405103
OnlineSecurity_No                          0.342637
TechSupport_No                             0.337281
InternetService_Fiber optic                0.308020
PaymentMethod_Electronic check             0.301919
OnlineBackup_No                            0.268005
DeviceProtection_No                        0.252481
MonthlyCharges                             0.193356
PaperlessBilling_Yes                       0.191825
Dependents_No                              0.164221
SeniorCitizen                              0.150889
Partner_No                                 0.150448
StreamingMovies_No                         0.130845
StreamingTV_No                             0.128916
StreamingTV_Yes                            0.063228
StreamingMovies_Yes                        0.061382
MultipleLines_Yes                          0.040102
PhoneService_Yes                           0.011942
gender_Female                              0.008612
gender_Male                               -0.008612
MultipleLines_No phone service            -0.011942
PhoneService_No                           -0.011942
MultipleLines_No                          -0.032569
DeviceProtection_Yes                      -0.066160
OnlineBackup_Yes                          -0.082255
PaymentMethod_Mailed check                -0.091683
PaymentMethod_Bank transfer (automatic)   -0.117937
InternetService_DSL                       -0.124214
PaymentMethod_Credit card (automatic)     -0.134302
Partner_Yes                               -0.150448
Dependents_Yes                            -0.164221
TechSupport_Yes                           -0.164674
OnlineSecurity_Yes                        -0.171226
Contract_One year                         -0.177820
PaperlessBilling_No                       -0.191825
TotalCharges                              -0.198324
DeviceProtection_No internet service      -0.227890
StreamingMovies_No internet service       -0.227890
InternetService_No                        -0.227890
OnlineSecurity_No internet service        -0.227890
StreamingTV_No internet service           -0.227890
TechSupport_No internet service           -0.227890
OnlineBackup_No internet service          -0.227890
Contract_Two year                         -0.302253
tenure                                    -0.352229
Name: Churn, dtype: float64
```

In [ ]:
```python
# Use barplot to plot the correlation more clearly
sns.set(style = 'whitegrid')
plt.figure(figsize = (10, 8))
tcc_dummy_corr['Churn'].sort_values(ascending = False)[1:].plot(kind = 'bar') # exclude "Churn" itself
plt.ylabel('Correlation with Churn')
plt.xlabel('Features')
plt.title('Correlation between features and label')
plt.show()
```



- The correlation coefficient of 0.4 between Churn and Contract Month-to-month suggests a positive relationship between the two. This implies that a user's likelihood of churning increases as they have a month-to-month contract instead of a long-term contract.
- On the other hand, a negative correlation (-0.35) between tenure and churn suggests that the longer a customer has been with the company, the less likely they are to churn.

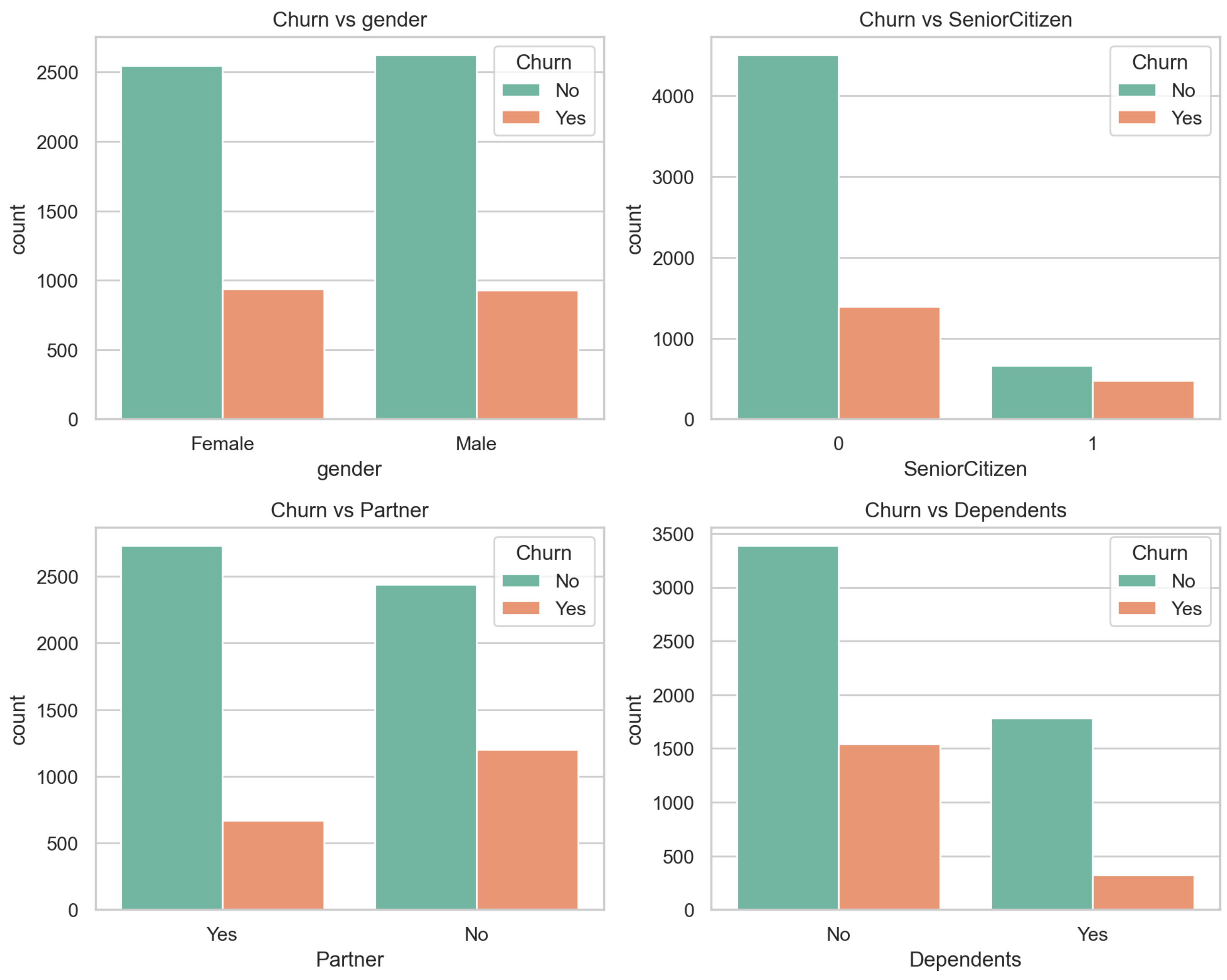
Distribution of categorical features vs label

Here categorical features can be divided into 3 categories:

1. **Demographic** feature: Gender, SeniorCitizen, Partner, Dependents
2. **Account** feature: Tenure, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges

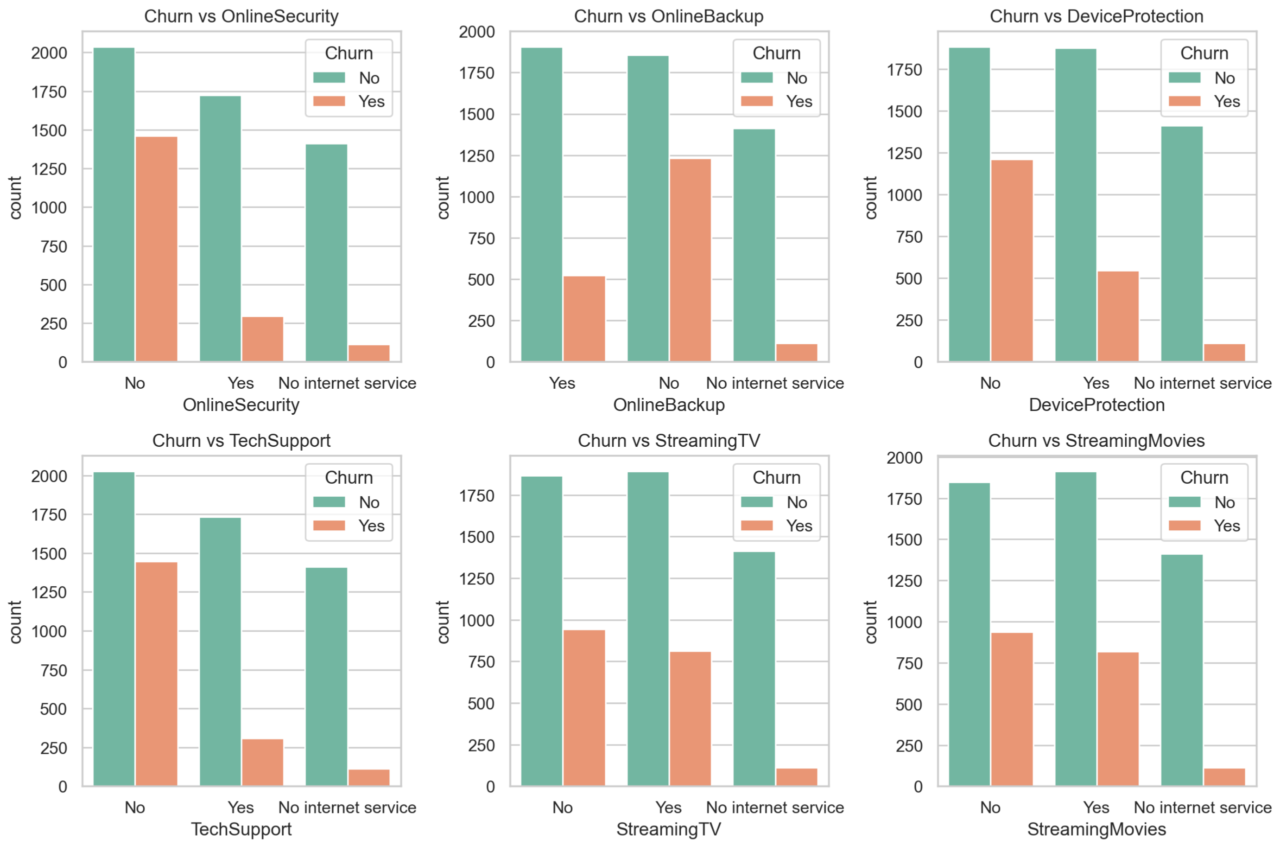
3. **Service** feature: PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies.

```
In [ ]: demographic = ["gender", "SeniorCitizen", "Partner", "Dependents"]

        fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (10, 8))
        for i in range(len(demographic)):
            sns.countplot(x = demographic[i], hue = 'Churn', data = tcc, palette = 'Set2', ax = axes[i//2, i%2])
            axes[i//2, i%2].set_title(f'Churn vs {demographic[i]}', fontsize = 12)
            plt.tight_layout()
```



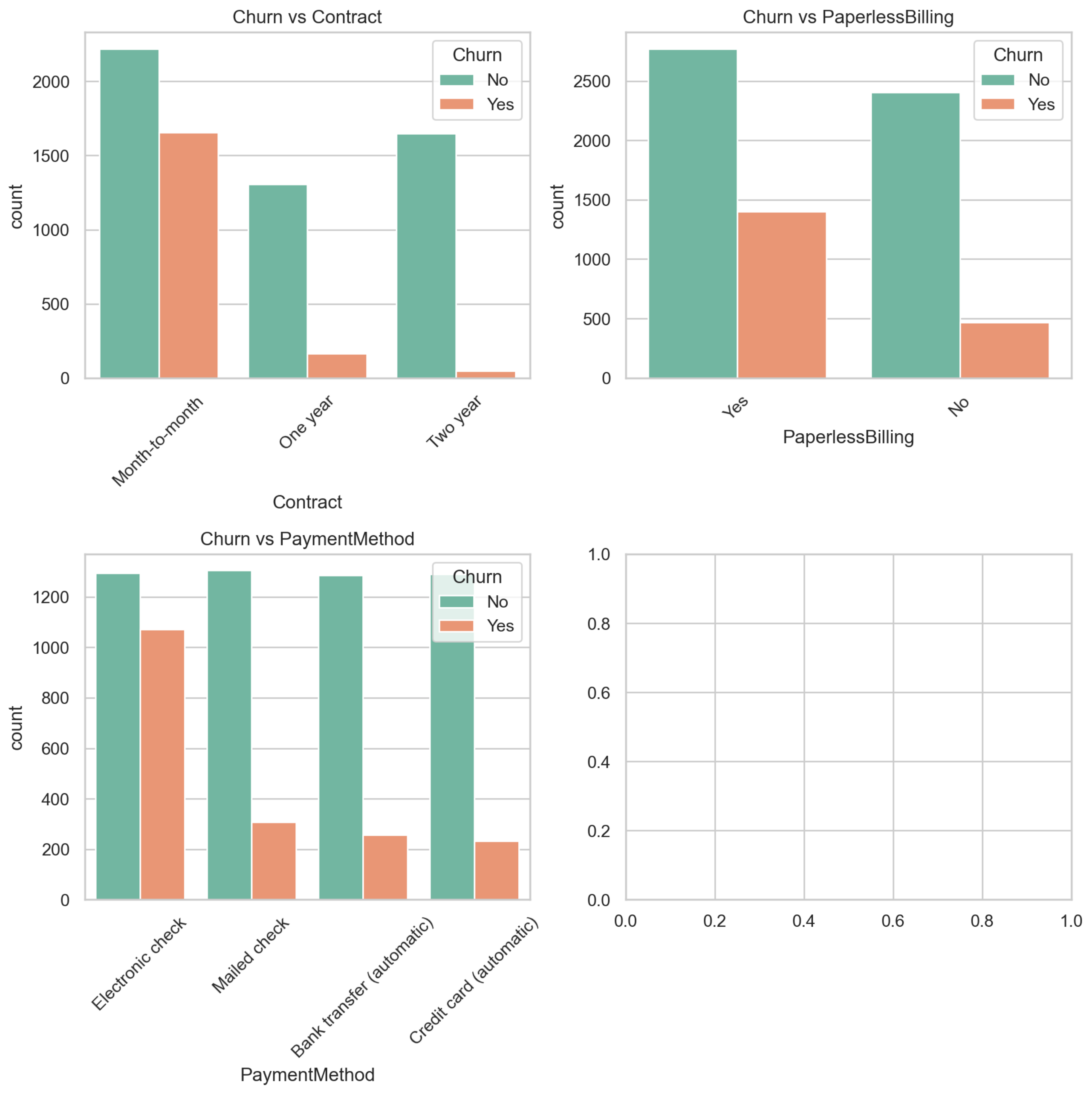
Churn rate for female/male is essentially the same, which means gender unlikely to have a significant impact on churn rate. However, older customers, unmarried users, and users who are not financially independent are more likely to churn. Therefore, when creating operational plans and strategies, it's important to pay special attention to these three user segments, as they are more likely to be at risk of churning.

```
In [ ]: service = ["OnlineSecurity", "OnlineBackup", "DeviceProtection", \
                   "TechSupport", "StreamingTV", "StreamingMovies"]

        fig, axes = plt.subplots(nrows = 2, ncols = 3, figsize = (12, 8))
        for i in range(len(service)):
            sns.countplot(x = service[i], hue = 'Churn', data = tcc, \
                          palette = 'Set2', ax = axes[i//3, i%3])
            axes[i//3, i%3].set_title(f'Churn vs {service[i]}', fontsize = 12)
            plt.tight_layout()
```



- Users are less likely to churn if they do not use the network service or if they use the network service but have several value-added services. However, users who use the network service but do not use other value-added services are more likely to churn.
- This suggests that the use of value-added services may play a role in customer loyalty. To encourage customers to purchase additional value-added services and enhance customer loyalty, the company should consider expanding the options for free use of value-added services.

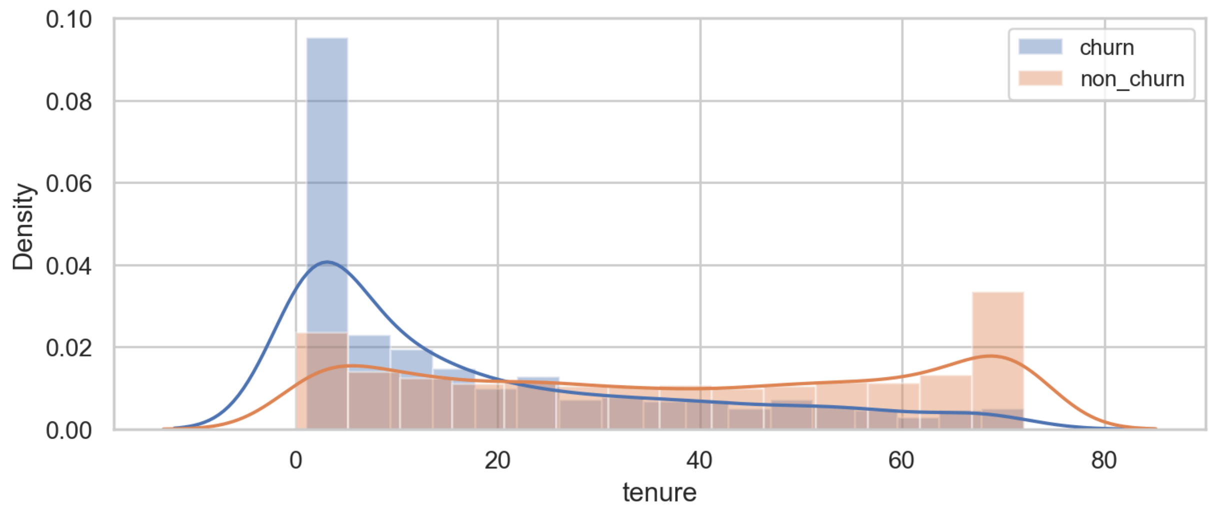
```
In [ ]: account = ["Contract", "PaperlessBilling", "PaymentMethod"]

        fig, axes = plt.subplots(nrows = 2, ncols = 2, figsize = (10, 10))
        for i in range(len(account)):
            sns.countplot(x = account[i], hue = 'Churn', data = tcc, \
                          palette = 'Set2', ax = axes[i//2, i%2])
            axes[i//2, i%2].set_title(f'Churn vs {account[i]}', fontsize = 12)
            axes[i//2, i%2].set_xticklabels(axes[i//2, i%2].get_xticklabels(), rotation = 45)
            plt.tight_layout()
```

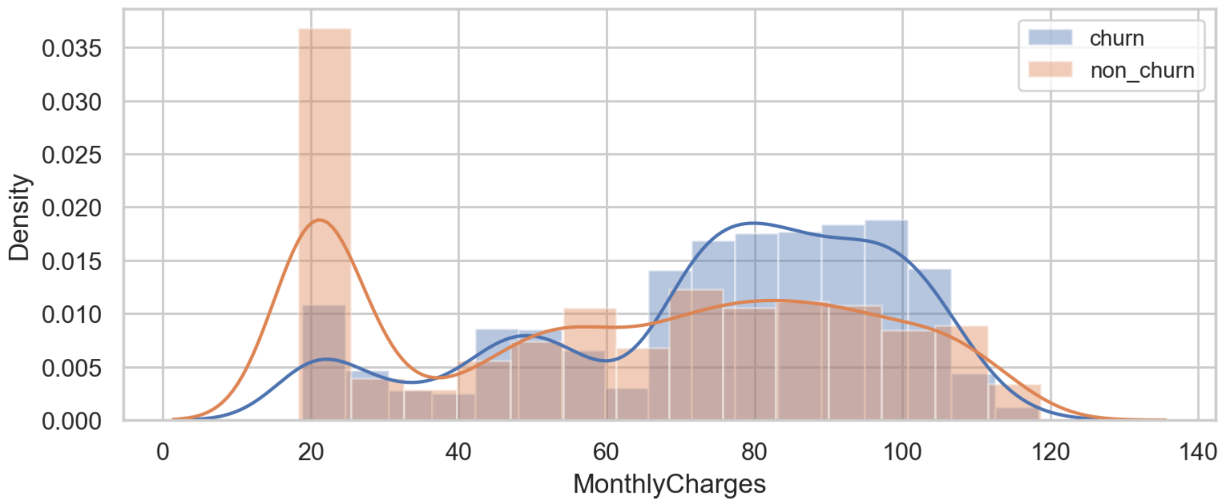


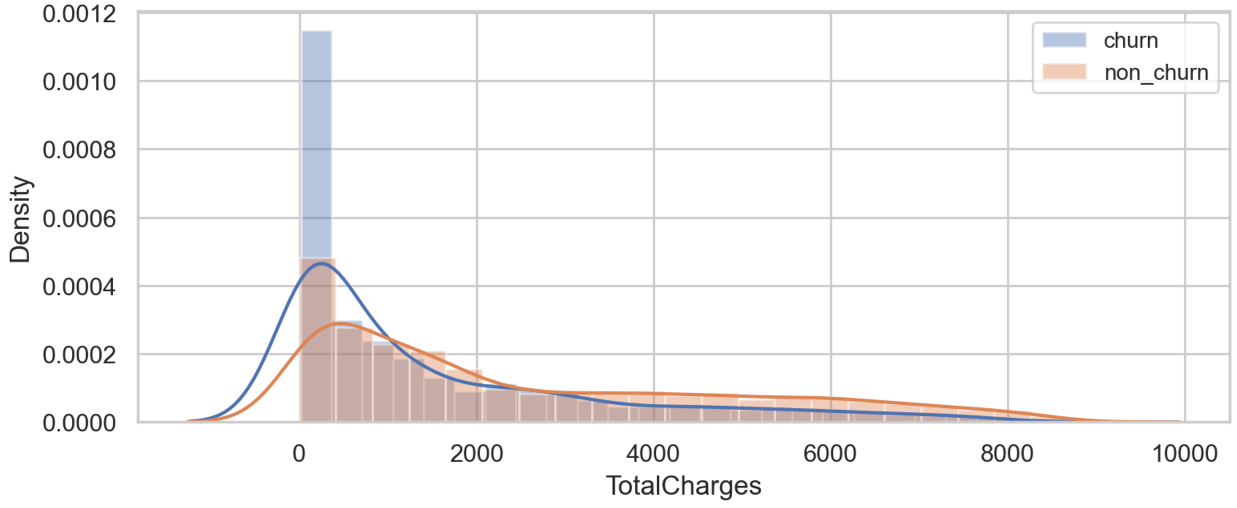
- Users who have signed one-time contracts are more likely to churn. Compared to other payment methods, users who pay online are also more likely to churn.
- To improve user retention, it may be necessary to encourage customers to sign long-term contracts more frequently. This can be done by offering discounts or gifts for signing long-term contracts, or by highlighting the benefits of long-term contracts.
- To increase satisfaction and reduce the likelihood of churn among online payment users, it is important to pay greater attention to their real product experiences.

Distribution of numeric features against label

```
In [ ]: numeric_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
        fig, axes = plt.subplots(ncols = 1, nrows = 3, figsize = (8, 10))
        for i in range(len(numeric_cols)):
            sns.distplot(tcc[tcc['Churn'] == 'Yes'][numeric_cols[i]], kde = True, ax = axes[i], label = 'churn')
            sns.distplot(tcc[tcc['Churn'] == 'No'][numeric_cols[i]], kde = True, ax = axes[i], label = 'non_churn')
            axes[i].legend(fontsize = 10)
            plt.tight_layout()
```







- Customers with fewer tenure are more likely to churn
- Higher total charge per month leads to higher customer churn rate

(d) Conclusion

- The analysis shows that customers with the following features are more likely to churn: older age, lack of use of online services, signing one-time contracts, shorter tenure, and higher total monthly charge.

- On the other hand, customers with the following features are less likely to churn: use of network services, use of value-added services such as online security and streaming.

To reduce churn rate, the following actions can be considered:

- Target retention efforts towards customers with shorter tenure, such as offering incentives for signing long-term contracts or highlighting the benefits of long-term contracts.
- Offer discounts or promotions to encourage customers to sign up for online services, which can help to increase their engagement with the company and reduce the likelihood of churn.
- Develop targeted retention programs for older customers, as they are more likely to churn. This can be done by offering special promotions or incentives, or by developing programs that cater to their specific needs and concerns.
- Monitor the total monthly charge of the customer and consider offering discounts or promotions for customers with high total monthly charge, as they are more likely to churn.
- Enhance the marketing and promotion of value-added services to raise awareness among customers and encourage customers to use services such as online security and streaming, as they can play a role in customer loyalty.