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

Maobin Guo

Netid: mg471

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

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, estimating probabilities through counting, 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, intepretation, 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 that is unfamiliar.

Probability and Statistics Theory

Note: for all assignments, write out all equations and math using markdown and LaTeX. For this assignment show ALL math work

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 :

For $\int_{-\infty}^{\infty} f(x)dx = \int_0^2 ax^2 = \frac{8}{3}a = 1$

Hence a = $\frac{3}{8}$

2

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

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

ANSWER

$\int \frac{1}{3}dx = \frac{x}{3} + c$

For $\int_0^3 \frac{1}{3}dx = 1$, Thus c = 0

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

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 $E(x) = \int_{-\infty}^{\infty} xf(x)dx$

For this distribution :

$E(x) = \int_0^3 \frac{x}{3}dx = \left[\frac{x^2}{6}\right]_0^3 = \frac{3}{2}$

expected value is $\frac{3}{2}$

(b) variance

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

For this distribution :

$$\begin{aligned} V(x) &= \int_0^3 \frac{1}{3}(x - E(x))^2 dx \\ &= \int_0^3 \frac{1}{3}(x - 1.5)^2 dx \\ &= \frac{1}{3}\left[\frac{1}{3}(x - 1.5)^3\right]_0^3 \\ &= 0.75 \end{aligned}$$

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
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

1. mean

$\mu = \frac{(x_0+x_1+x_2+x_3+x_4)}{N} = \frac{(2+3+10-1-1)}{5} = \frac{13}{5} = 2.6$

2. median

Sorted Data: -1, -1, 2, 3, 10
median = 2 ; because it is in the middel of above array

3. variance

$$\begin{aligned} variance &= \sum(x_i - \mu)^2 p(x_i) \\ &= \frac{1}{5}((2 - \frac{13}{5})^2 + (3 - \frac{13}{5})^2 + (10 - \frac{13}{5})^2 + (-1 - \frac{13}{5})^2) + (-1 - \frac{13}{5})^2) \\ &= \frac{406}{25} = 16.24 \end{aligned}$$

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)

It's a 7-place replaceable permutation
$26^3 \times 10^4 = 175760000$

(b)

It's a full permutation of 9
$9! = 362880$

(c)

There are 2 steps:

1. Select 5 player from 9 player, there are $\binom{9}{5}$ possible methods
2. Order the 5 selected player, it's a full permutation so there are **5!** possible methods

Total method:

$\binom{9}{5} \times 5! = \frac{9!}{(9-5)!\times 5!} \times 5! = 9 \times 8 \times 7 \times 6 \times 5 = 15120$

(d)

Choose 3 from 26

$\binom{26}{3} = \frac{26!}{(26-3)!3!} = 2600$

Linear Algebra

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 "∘".

ANSWER

In [1]:

```
import numpy as np
A = np.matrix([(1,2,3),
               (2,4,5),
               (3,5,6)])
b = np.array([[-1,3,8]])
c = np.array([[4,-3,6]])
I = np.matrix([(1,0,0),
               (0,1,0),
               (0,0,1)])
```

1. $\mathbf{AA}$

In [7]:

```
print('AA = \n {}'.format(np.matmul(A,A)))
```

```
AA =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]
```

2. $\mathbf{AA}^T$

In [12]:

```
print('A(A^T) = \n {}'.format(np.matmul(A,A.T)))
```

```
A(A^T) =
 [[14 25 31]
 [25 45 56]
 [31 56 70]]
```

3. $\mathbf{Ab}$

In [13]:

```
print('Ab = \n {}'.format(np.matmul(A,b)))
```

```
Ab =
 [[29 50 60]]
```

4. $\mathbf{Ab}^T$

The operation will fail because A is a 3×3 matrix while b^T is 1×3 matrix

5. $\mathbf{bA}$

The operation will fail because b^T is 3×1 matrix while A is a 3×3 matrix while

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

In [23]:

```
print('b^TA = \n {}'.format(np.matmul(b.T,A)))
```

```
b^TA =
 [[29 50 60]]
```

7. $\mathbf{bb}$

The operation will fail because 3×1 matrix can not multiply another 3×1 matrix

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

In [17]:

```
print('b^Tb = \n {}'.format(np.matmul(b.T,b)))
```

```
b^Tb =
 74
```

9. $\mathbf{bb}^T$

In [22]:

```
print('bb^T = \n {}'.format(np.matmul(np.matrix([[(-1),(3),(8)]]).T,np.matrix([[(-1),(3),(8)]]))))
```

```
bb^T =
 [[ 1 -3 -8]
 [-3  9 24]
 [-8 24 64]]
```

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

The operation will fail because b is a 3×1 matrix while c^T is 1×3 matrix

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

The operation will fail because b is a 1×3 matrix while b^T is 1×3 matrix

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

In [25]:

```
print('A^-1b = \n {}'.format(np.matmul(np.linalg.inv(A), b)))
```

```
A^-1b =
 [[ 6.  4. -5.]]
```

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

In [26]:

```
print('A.A = \n {}'.format(np.multiply(A, A)))
```

```
A.A =
 [[ 1  4  9]
 [ 4 16 25]
 [ 9 25 36]]
```

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

In [27]:

```
print('b.c = \n {}'.format(np.multiply(b, c)))
```

```
b.c =
 [-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. 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 $\mathbf{A}$ 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.

ANSWER

1. Eigenvalues and corresponding eigenvectors
$det|\mathbf{A} - \lambda\mathbf{I}| = 0$
$\Rightarrow (1-\lambda)det\begin{bmatrix} 4-\lambda & 5 \\ 5 & 6-\lambda \end{bmatrix} - 2det\begin{bmatrix} 2 & 5 \\ 3 & 6-\lambda \end{bmatrix} + 3det\begin{bmatrix} 2 & 4-\lambda \\ 3 & 5 \end{bmatrix} = 0$
$\Rightarrow (1-\lambda)((4-\lambda)(6-\lambda) - 25) - 2(2(6-\lambda) - 15) + 3(10 - 3(4-\lambda)) = 0$
$\Rightarrow -\lambda^3 + 11\lambda^2 + 4\lambda - 1 = 0$

$\lambda_0 = 11.3448$
$\lambda_1 = -0.5157$
$\lambda_2 = 0.1709$

$v_0 = \begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix}$

$v_1 = \begin{bmatrix} -0.737 \\ -0.328 \\ 0.591 \end{bmatrix}$

$v_2 = \begin{bmatrix} 0.591 \\ -0.737 \\ 0.328 \end{bmatrix}$

In [39]:

```
import numpy as np
from numpy import linalg

np.set_printoptions(precision=4)

eival, eivec = linalg.eig(A)

rank = linalg.matrix_rank(A)

print("There are {} pairs of eigenvalues and eigenvector".format(rank))

for i in range(rank):
    print("{}.)".format(i))
    print(eivec[:,i])
    print("{:.4f}".format(eival[i]))
```

```
There are 3 pairs of eigenvalues and eigenvector
0.)
[[-0.328]
 [-0.591]
 [-0.737]]
11.3448
1.)
[[-0.737]
 [-0.328]
 [ 0.591]]
-0.5157
2.)
[[ 0.591]
 [-0.737]
 [ 0.328]]
0.1709
```

2. Choose one of the eigenvector/eigenvalue pairs, v and λ, and show that $\mathbf{Av} = \lambda\mathbf{v}$. Also show that this relationship extends to higher orders: $\mathbf{AAv} = \lambda^2\mathbf{v}$
$\lambda_0 = 11.3448$

$\mathbf{v_0} = \begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix}$

$\mathbf{Av_0} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix} = \begin{bmatrix} -3.720 \\ -6.705 \\ -8.361 \end{bmatrix}$

$\lambda_0\mathbf{v_0} = 11.3448\begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix} = \begin{bmatrix} -3.720 \\ -6.705 \\ -8.361 \end{bmatrix}$

$\mathbf{AAv_0} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}\begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix} = \begin{bmatrix} -42.213 \\ -76.066 \\ -94.852 \end{bmatrix}$

$\lambda_0{}^2\mathbf{v_0} = 11.3448^2\begin{bmatrix} -0.328 \\ -0.591 \\ -0.737 \end{bmatrix} = \begin{bmatrix} -42.213 \\ -76.066 \\ -94.852 \end{bmatrix}$

$\mathbf{Av}$

In [45]:

```
print(np.dot(A, eivec[:, 0]))
```

```
[[-3.7209]
 [-6.7049]
 [-8.3609]]
```

$\lambda\mathbf{v}$

In [46]:

```
print("A*v:")
print(eivec[:, 0]*eival[0])
```

```
A*v:
[[-3.7209]
 [-6.7049]
 [-8.3609]]
```

So : $\mathbf{Av} = \lambda\mathbf{v}$

$\mathbf{AAv}$

In [48]:

```
print(np.dot(np.dot(A, A), eivec[:, 0]))
```

```
[[-42.2133]
 [-76.0657]
 [-94.8524]]
```

$\lambda^2\mathbf{v}$

In [55]:

```
print(eivec[:, 0]*eival[0]*eival[0])
```

```
[[-42.2133]
 [-76.0657]
 [-94.8524]]
```

So: $\mathbf{AAv} = \lambda^2\mathbf{v}$

3. Show that the eigenvectors are orthogonal to one another (e.g. their inner product is zero). This is true for real, symmetric matrices.

$\mathbf{v_0v_1} = [-0.328 \ -0.591 \ -0.737]\begin{bmatrix} -0.737 \\ -0.328 \\ 0.591 \end{bmatrix} = 0$

$\mathbf{v_1v_2} = [-0.328 \ -0.591 \ -0.737]\begin{bmatrix} 0.591 \\ -0.737 \\ 0.328 \end{bmatrix} = 0$

$\mathbf{v_2v_0} = [-0.737 \ -0.328 \ 0.591]\begin{bmatrix} 0.591 \\ -0.737 \\ 0.328 \end{bmatrix} = 0$

In [68]:

```
print('v0 * v1 is zero ', np.isclose(0, np.dot(eivec[:, 0].T, eivec[:, 1]))[0,0] )
print('v1 * v2 is zero ', np.isclose(0, np.dot(eivec[:, 1].T, eivec[:, 2]))[0,0] )
print('v2 * v0 is zero ', np.isclose(0, np.dot(eivec[:, 2].T, eivec[:, 0]))[0,0] )
```

```
v0 * v1 is zero  True
v1 * v2 is zero  True
v2 * v0 is zero  True
```

Since all inner product is 0, so the eigenvectors are orthogonal to one another.

Numerical Programming

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 showing the amount of energy produced by each fuel for the plant.

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

ANSWER

In [4]:

```
import os
import pandas as pd

df = pd.read_excel('./data/egrid2016.xlsx', skiprows=[0], engine='openpyxl')
df.head()
```

Out[4]:

	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

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

In [14]:

```
df[df["PLNGENAN"] == df["PLNGENAN"].max()][["PSTATABB", "PNAME"]]
```

Out[14]:

	PSTATABB	PNAME
390	AZ	Palo Verde

Palo Verde plant generated the most energy

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

In [15]:

```
df[df["LAT"] == df["LAT"].max()][["PSTATABB", "PNAME"]]
```

Out[15]:

	PSTATABB	PNAME
11	AK	Barrow

Barrow is the northern-most power plant

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

In [ ]:

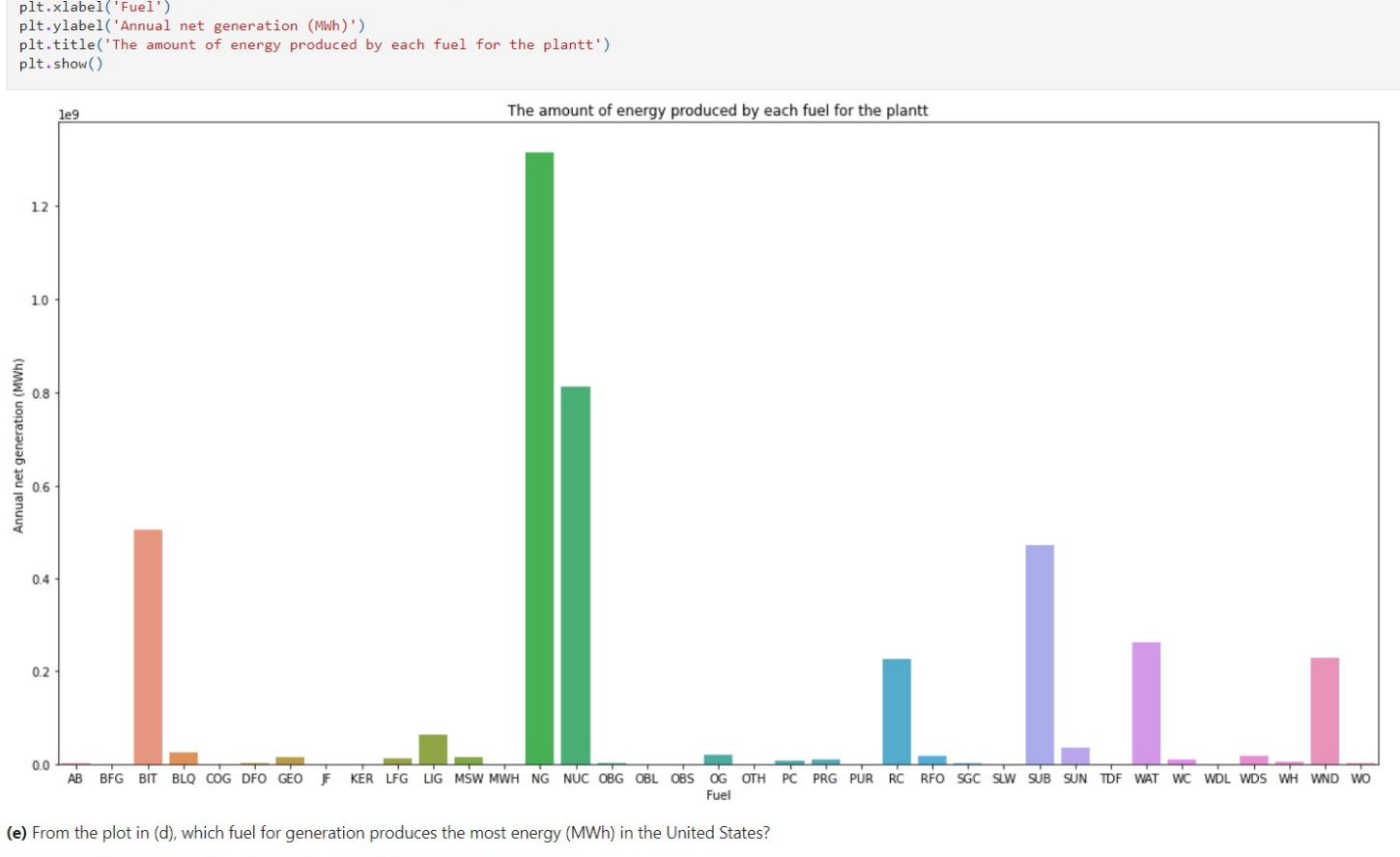
```
df[df["LAT"] == df["LAT"].max()][["PSTATABB", "PNAME"]]
```

The northern-most power plant is in **AK (Alaska)**

(d) Plot a bar plot showing the amount of energy produced by each fuel for the plant.

In [41]:

```
import seaborn as sns
import matplotlib.pyplot as plt

df_fule = df[['PLPRMFL','PLNGENAN']]
df_fule.dropna()
df_fule = df_fule.groupby('PLPRMFL').sum().reset_index()

plt.figure(figsize=(20,10))
sns.barplot(x='PLPRMFL', y='PLNGENAN', data=df_fule, ci=None)
plt.xlabel('Fuel')
plt.ylabel('Annual net generation (MWh)')
plt.title('The amount of energy produced by each fuel for the plantt')
plt.show()
```



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

The **Natural Gas (NG)** produces the most energy (MWh) in the United States.

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

In [47]:

```
import numpy as np
import time

# generate data
data = np.random.randn(10000000)

# non-vectorized
sqar = 0
t0 = time.time()
for n in data:
    sqar += n**2
t1 = time.time()
time_non_vect = t1 - t0

# vectorized
t0 = time.time()
ans = np.dot(data, data)
t1 = time.time()
time_vect = t1 - t0
```

In [55]:

```
print('Time [sec] (non-vectorized): {:.4f} seconds'.format(time_non_vect))
print('Time [sec] (vectorized): {:.4f} second'.format(time_vect))
print('The vectorized method is {} times faster'.format(int(time_non_vect/time_vect)))
```

```
Time [sec] (non-vectorized): 5.3198 seconds
Time [sec] (vectorized): 0.0030 second
The vectorized method is 1768 times faster
```

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

In [52]:

```
import numpy as np
import time
import matplotlib.pyplot as plt

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
```

In [54]:

```
# Vectorized implementation
t0 = time.time()
X, Y = np.meshgrid(xvalues, yvalues)
f = X ** 2 - 2 * Y**2
f_thresholded = f > thresh
t1 = time.time()
time_vectorized = t1 - t0
```

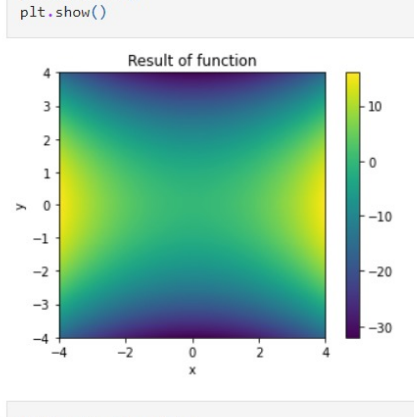
In [57]:

```
# Vectorization speed performance results
print('Time [sec] (non-vectorized): {:.4f} seconds'.format(time_nonvectorized))
print('Time [sec] (vectorized): {:.4f} second'.format(time_vectorized))
print('The vectorized method is {} times faster'.format(int(time_nonvectorized/time_vectorized)))
```

```
Time [sec] (non-vectorized): 8.1622 seconds
Time [sec] (vectorized): 0.0788 second
The vectorized method is 103 times faster
```

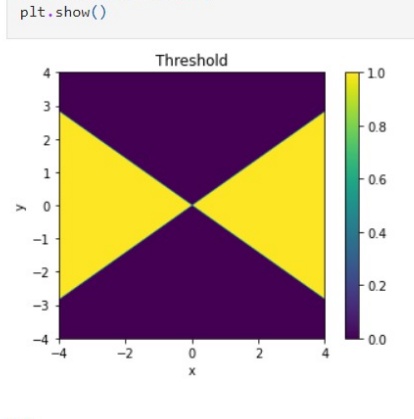
In [58]:

```
# Plot the function f(x,y)
import matplotlib.pyplot as plt

plt.imshow(f, extent = [-4, 4, -4, 4])
plt.xlabel('x')
plt.ylabel('y')
plt.title("Result of function")
plt.colorbar()
plt.show()
```



In [43]:

```
# Plot the threshold
plt.imshow(f_thresholded, extent = [-4, 4, -4, 4])
plt.colorbar()
plt.xlabel('x')
plt.ylabel('y')
plt.title('Threshold')
plt.show()
```



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

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(471)
```

In [66]:

```
##### Question 1
x = np.random.normal(2, 1, 10000)
```

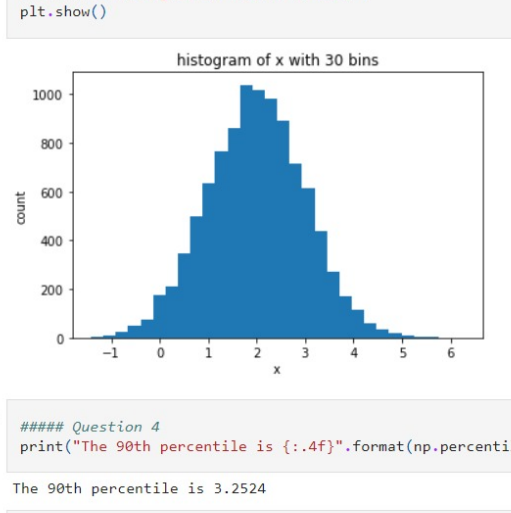
In [68]:

```
##### Question 2
print("mean of x is {:.4f}".format(np.mean(x)))
print("standard deviation of x is {:.4f}".format(np.std(x)))
```

```
mean of x is 1.9837
standard deviation of x is 0.9987
```

In [72]:

```
##### Question 3
plt.hist(x, bins=30)
plt.xlabel('x')
plt.ylabel('count')
plt.title('histogram of x with 30 bins')
plt.show()
```



In [73]:

```
##### Question 4
print("The 90th percentile is {:.4f}".format(np.percentile(x, 90)))
```

```
The 90th percentile is 3.2524
```

In [74]:

```
##### Question 5
print("The 99th percentile is {:.4f}".format(np.percentile(x, 99)))
```

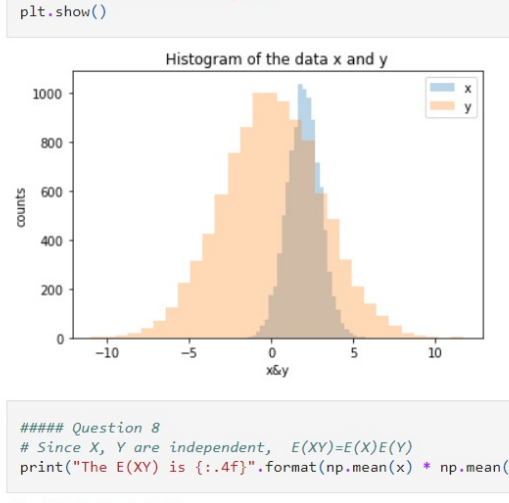
```
The 99th percentile is 4.2996
```

In [79]:

```
##### Question 6
y = np.random.normal(0, 3, 10000)
```

In [81]:

```
##### Question 7
plt.hist(x, bins=30, alpha=0.3, label='x')
plt.hist(y, bins=30, alpha=0.3, label='y')
plt.title('Histogram of the data x and y')
plt.xlabel('x&y')
plt.ylabel('counts')
plt.legend(loc='upper right')
plt.show()
```



In [89]:

```
##### Question 8
# Since X, Y are independent,  E[XY]=E[X]E[Y]
print("The E[XY] is {:.4f}".format(np.mean(x) * np.mean(y)))
```

```
The E[XY] is 0.0059
```

Version Control via Git

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 this 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, ***[Maobin Guo]****, affirm that I have* ***[completed the above tutorial / I have previous experience that covers all the content in this tutorial]***

Exploratory Data Analysis

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.
4. Professionalism: Was this work done in a way that exhibits professionalism through clarity, organization, high quality figures and plots, and meaningful descriptions?

ANSWER

Score Wines

Introduction

This report analyzes the impact of physicochemical properties on wine grades by examining the association between the properties and the wine flavor scores. Establishing an objective method to evaluate wine quality has always been a hot issue for the industry. Such an objective evaluation is valuable because it can help winemakers produce better wines and help wine merchants set suitable prices for thousands of wines more wisely. Furthermore, consumers can also have a reliable approach to choosing the best wines. This article would have a glance on the data which contains thousands wine's physicochemical properties and their flavor score given by tasters, and try to find out some insights.

Data Description

The data used in this paper was created by Cortez et al. (1998). It includes two datasets related to red and white variants of the Portuguese "Vinho Verde" wine. The data is acquired from UCI Machine Learning Repository: UCI Data.

Feature Description

Input variables (based on physicochemical tests):

- FAC : Fixed acidity (g(tartaric acid)/dm3)
- VAC : Volatile acidity (g(acetic acid)/dm3)
- CAC : Citric acid (g/dm3)
- RS : Residual sugar (g/dm3)
- CHOL : Chlorides (g(sodium chloride)/dm3)
- FSD : Free sulfur dioxide (mg/dm3)
- TSD : Total sulfur dioxide (mg/dm3)
- DEN : Density (g/cm3)
- PH : pH
- SUL : Sulphates (g(potassium sulphate)/dm3)
- ALC : Alcohol (vol.%)
- TYPE : Wine type (red wine| white wine)

Output variable

- SCORE : (score between 0 and 10)

In [194]:

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Loading data
df = pd.read_csv('./data/wine.csv')
```

Data Checking

In [195]:

```
df.head()
```

Out[195]:

	FAC	VAC	CAC	RS	CHOL	FSD	TSD	DEN	PH	SUL	ALC	TYPE	SCORE
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	red	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	red	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	red	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	red	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	red	5

Checking Missing Value

In [196]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   FAC     6497 non-null   float64
 1   VAC     6497 non-null   float64
 2   CAC     6497 non-null   float64
 3   RS      6497 non-null   float64
 4   CHOL    6497 non-null   float64
 5   FSD     6497 non-null   float64
 6   TSD     6497 non-null   float64
 7   DEN     6497 non-null   float64
 8   PH      6497 non-null   float64
 9   SUL     6497 non-null   float64
 10  ALC     6497 non-null   float64
 11  TYPE    6497 non-null   object
 12  SCORE   6497 non-null   int64
dtypes: float64(11), int64(1), object(1)
memory usage: 660.0+ KB
```

The code indicats that there is no missing value in this dataset.

Check basic information of every features

In [3]:

```
df.describe()
```

Out[3]:

	FAC	VAC	CAC	RS	CHOL	FSD	TSD	DEN	PH	SUL	ALC	SCORE
count	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000	6497.000000
mean	7.215307	0.339666	0.318633	5.443235	0.056034	30.525319	115.744574	0.994697	3.218501	0.531268	10.491801	5.818378
std	1.296434	0.164636	0.145318	4.757804	0.035034	17.749400	56.521855	0.002999	0.160787	0.148806	1.192712	0.873255
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000	0.220000	8.000000	3.000000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000	0.430000	9.500000	5.000000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000	0.510000	10.300000	6.000000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000	0.600000	11.300000	6.000000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000	2.000000	14.900000	9.000000

In [195]:

```
df["TYPE"].value_counts()
```

Out[195]:

```
white    4898
red      1599
Name: TYPE, dtype: int64
```

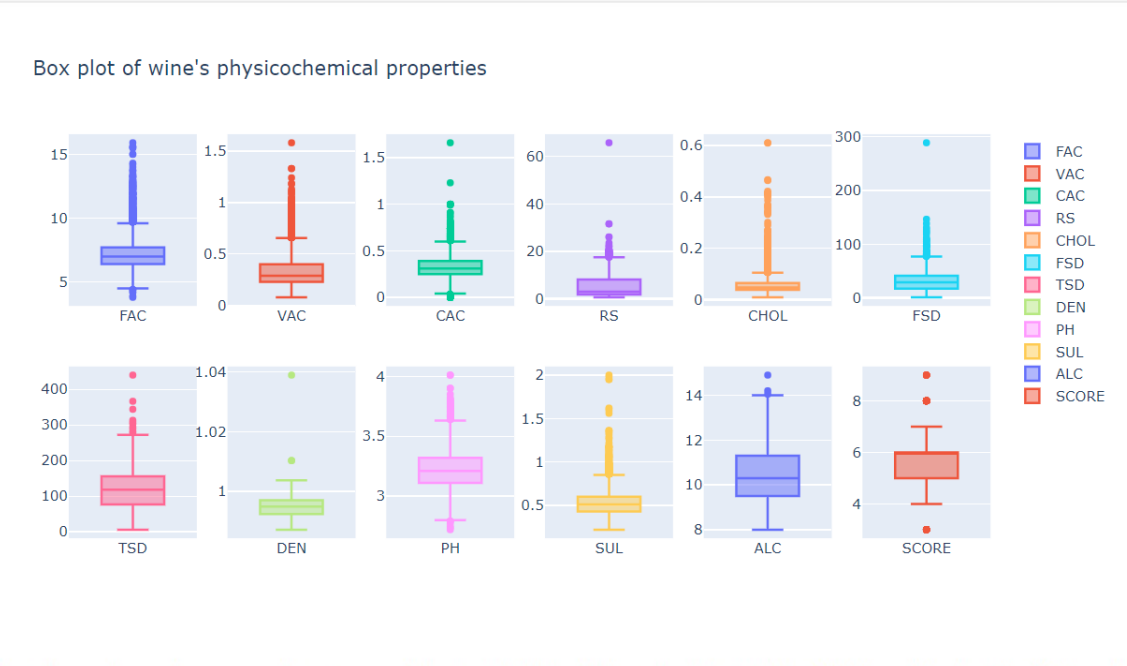
Summary

This dataset consists of two kind of wines: white wine containing 4898 observations and red wine containing 1599 observations. The two datasets share the same structure and attributes, and there are no missing fields. The predictor variables include: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol; and the response variable is score. All variables except wine type are continuous.

Exploratory Data Analysis

In [55]:

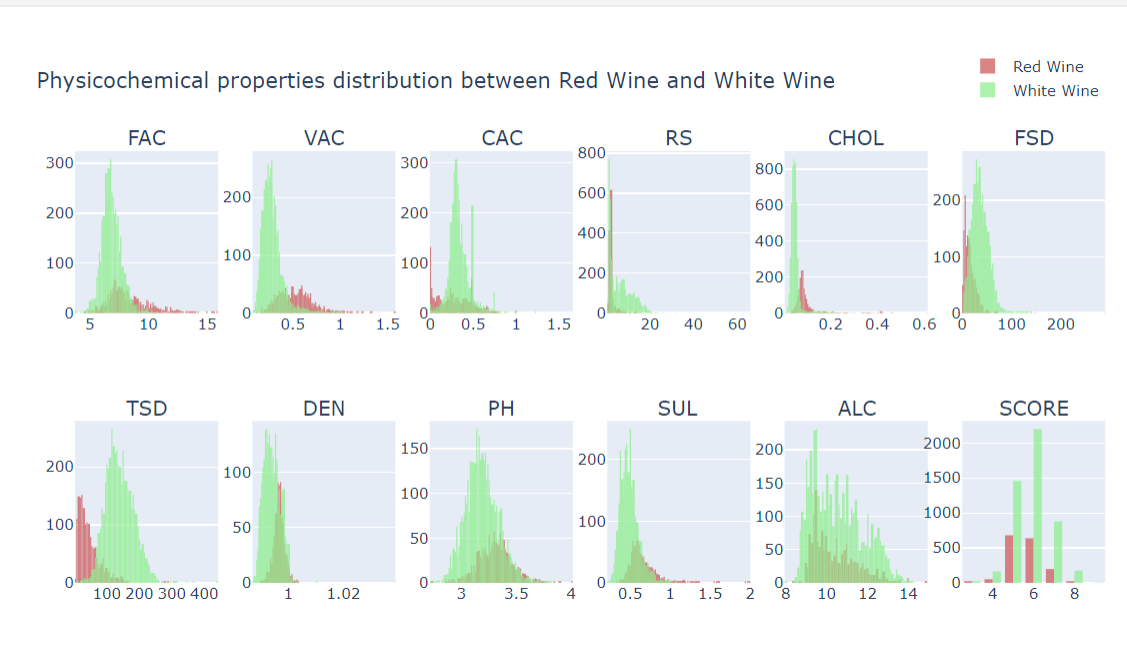
```
from plotly.subplots import make_subplots

FIG_COLUMN_SIZE = 6

fig = make_subplots(rows=2, cols=FIG_COLUMN_SIZE)
index = 0

for c in df.columns:

    if c in ['TYPE']:
        continue

    row, col = divmod(index, FIG_COLUMN_SIZE)
    fig.add_trace(go.Box(y=df[c], name=c), row=row+1, col=col+1)
    index += 1

fig.update_layout(title_text="Box plot of wine's physicochemical properties")

fig.show()
```



According to the boxplot, many features have many outliers, for example, fixed acidity (FAC), volatile acidity (VAC), chlorides (CHOL), free sulfur dioxide (FSD), PH, and sulphate(SUL). On the contrary, alcohol(ALC), density(DEN), and score have less outlier. There are some extraordinary outliers for density and score. Finding out the root cause of these outliers is interesting and necessary because they may have high leverage and greatly influence the final model. To quest why some wines got extra high/low score may provide some valuable sights of wines quality.

In [132]:

```
FIG_COLUMN_SIZE = 6

fig = make_subplots(rows=2,
                    cols=FIG_COLUMN_SIZE,
                    subplot_titles= list(filter(lambda x: x != "TYPE", df.columns)))
index = 0

for c in df.columns:

    if c in ['TYPE']:
        continue
    row, col = divmod(index, FIG_COLUMN_SIZE)

    #fig.add_trace(go.Box(y=df[c], name=c), row=1, col=1)
    index += 1

    if index == 1:
        fig.add_trace(go.Histogram(x=df[df["TYPE"] == "red"][c], marker_color='#CD5C5C', legendgroup='g1', opacity=0.75, name='Red Wine'), row=1, col=1)
        fig.add_trace(go.Histogram(x=df[df["TYPE"] == "white"][c], marker_color='#90EE90', legendgroup='g2', opacity=0.75, name='White Wine'), row=1, col=1)
    else:
        fig.add_trace(go.Histogram(x=df[df["TYPE"] == "red"][c], marker_color='#CD5C5C', showlegend=False, legendgroup='g1', opacity=0.75, name='Red Wine'), row=1, col=1
        fig.add_trace(go.Histogram(x=df[df["TYPE"] == "white"][c], marker_color='#90EE90', showlegend=False, legendgroup='g2', opacity=0.75, name='White Wine'), row=1, c

fig.update_layout(title_text="Physicochemical properties distribution between Red Wine and White Wine")

fig.update_layout(legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.1,
    xanchor="right",
    x=1
))

fig.show()
```

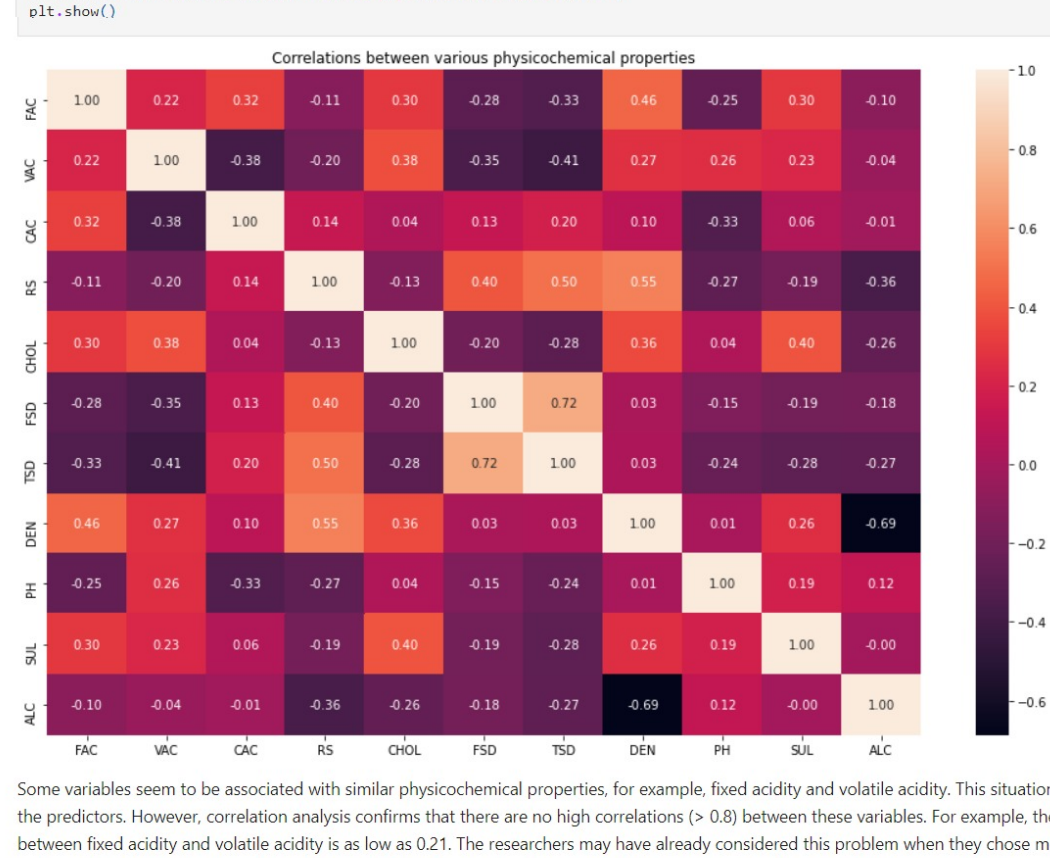


The graphs are the histograms of various features. For each feature, white wine and red wine data are plotted with different colors and displayed in one graph. The overlapped graph can help people figure out the difference of distribution on every physicochemical property. For some properties, for example, PH value, residual sugar (RS), and alcohol, the two types of wines' distributions are similar. However, the distributions on other properties are diverse, especially on total sulfur dioxide (TSD). The score is also noticeable; it shows that white wines are more likely to get a higher score than red wines. These differences suggest that a property may play different roles in white wine and red wines. Hence, the model build on these features should treat them in different ways according to wine types.

Another thing to notice is that some distributions are highly skewed, for example, residual sugar (RS) and free sulfur dioxide (FSD). Some models may require transformation on these features to make them normal.

In [134]:

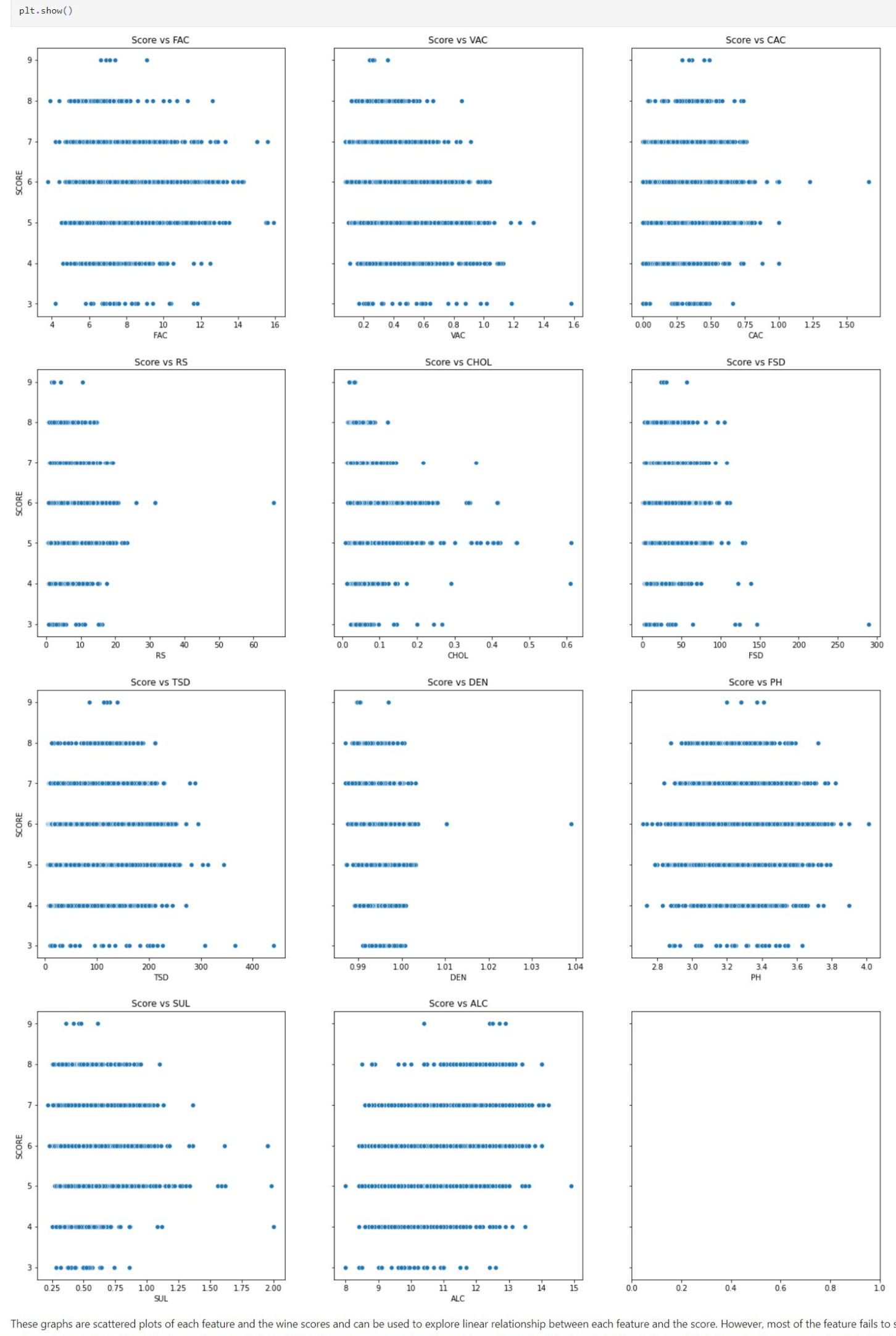
```
plt.figure(figsize=(16,10))
sns.heatmap(df_ps.corr(),fmt='.2f', annot=True)
plt.title('Correlations between various physicochemical properties')
plt.show()
```



Some variables seem to be associated with similar physicochemical properties, for example, fixed acidity and volatile acidity. This situation may raise concerns about the high correlation between the predictors. However, correlation analysis confirms that there are no high correlations (> 0.8) between these variables. For example, the absolute correlation coefficient value shows correlation between fixed acidity and volatile acidity is as low as 0.21. The researchers may have already considered this problem when they chose measurement indicators.

In [187]:

```
fig, axes = plt.subplots(4, 3, figsize=(20, 30), sharey=True)


for i, c in enumerate(df_ps.columns):

    if i == 11:
        break

    row, col = divmod(i, 3)
    sns.scatterplot(ax  = axes[row, col],
                    data = df,
                    x=c,
                    y="SCORE")

    axes[row, col].set_title('Score vs {}'.format(c))


plt.show()
```



These graphs are scattered plots of each feature and the wine scores and can be used to explore linear relationship between each feature and the score. However, most of the feature fails to show a strong linear relationship with wine grades except alcohol (ALC). There is an apparent pattern in the scatter plot of alcohol that high alcohol positively impacts wine score. Fortunately, the two wines' distributions on alcohol are very similar, suggesting that alcohol may be a universal factor for all wines. This finding should be verified in further analysis. Other features fail to shows an obvious pattern with the score. It is not easy to build a model to predict wine grades based on these features. Some non-linear models may be involved to get better performance.

Conclusion

The data is clean and ready for analysis. However, some feature's distribution is highly skewed and has a lot of outliers. The problem should be taken care of for some models. Correlation is not a problem for this data. Aside from alcohol, there is no obvious linear relationship between these features and wine scores, which indicates it is not easy to predict wine grades with these physicochemical properties.