NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

PRACTICAL FILE

SUBJECT: COMPUTATIONAL DATA SCIENCE
CODE: INITE68

By:
Dhruv Kumar Singh (2021UIN3324)
Vinayak Kapila (2021UIN3319)
Gaurav Kumar (2021UIN3312)

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

PRACTICAL FILE

SUBJECT: COMPUTATIONAL DATA SCIENCE
CODE: INITE68

By:
Dhruv Kumar Singh (2021UIN3324)
Vinayak Kapila (2021UIN3319)
Gaurav Kumar (2021UIN3312)

NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

PRACTICAL FILE

SUBJECT: COMPUTATIONAL DATA SCIENCE
CODE: INITE68

By:
Dhruv Kumar Singh (2021UIN3324)
Vinayak Kapila (2021UIN3319)
Gaurav Kumar (2021UIN3312)

1. Download and study any two databases and write a description of each database. Write a code in python to access the datasets using GPU.

a) Heart Disease dataset

The heart disease dataset [1] is a prediction dataset containing 302 instances and 74 attributes. It is available on UCI Machine Learning repository website, and is used for the task of classification. For practical purposes, a subset of 14 attributes out of the 74attributes are used. The target classes are: 1 for the presence of heart disease in a patient, while 0 is for no heart disease.

The data has been collected from 4 sources:

- Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
- University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
- University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
- V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

The 14 attributes are described as follows:

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholesterol in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by fluoroscopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversible defect

This dataset has been widely used in machine learning research, particularly in the areas of classification and predictive modeling. The goal of most studies using this dataset is to develop models that can accurately predict whether a patient has heart disease based on their demographic and medical information. The UCI Heart Diseasedataset is often used as a benchmark for evaluating the performance of different machine learning algorithms.

[1]:
```
!pip install cudf
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import cudf
import os
```

```
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: cudf in /opt/conda/lib/python3.8/site-packages (22.4.0a0+306.g0cb75a4913)
Requirement already satisfied: numpy in /opt/conda/lib/python3.8/site-packages (from cudf) (1.22.3)
Requirement already satisfied: nvtx>=0.2.1 in /opt/conda/lib/python3.8/site-packages (from cudf) (0.2.4)
Requirement already satisfied: cachetools in /opt/conda/lib/python3.8/site-packages (from cudf) (5.0.0)
Requirement already satisfied: fsspec>=0.6.0 in /opt/conda/lib/python3.8/site-packages (from cudf) (2022.3.0)
Requirement already satisfied: pandas<1.4.0dev0,>=1.0 in /opt/conda/lib/python3.8/site-packages (from cudf) (1.3.5)
Requirement already satisfied: cupy-cuda115 in /opt/conda/lib/python3.8/site-packages (from cudf) (9.6.0)
Requirement already satisfied: protobuf in /opt/conda/lib/python3.8/site-packages (from cudf) (3.20.1)
Requirement already satisfied: numba>=0.53.1 in /opt/conda/lib/python3.8/site-packages (from cudf) (0.53.1)
Requirement already satisfied: Cython<0.30,>=0.29 in /opt/conda/lib/python3.8/site-packages (from cudf) (0.29.28)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.8/site-packages (from cudf) (4.2.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.8/site-packages (from cudf) (21.3)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.8/site-packages (from numba>=0.53.1->cudf) (59.5.0)
Requirement already satisfied: llvmlite<0.37,>=0.36.0rc1 in /opt/conda/lib/python3.8/site-packages (from numba>=0.53.1->cudf) (0.36.0)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.8/site-packages (from pandas<1.4.0dev0,>=1.0->cudf) (2022.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.8/site-packages (from pandas<1.4.0dev0,>=1.0->cudf) (2.8.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.8/site-packages (from python-dateutil>=2.7.3->pandas<1.4.0dev0,>=1.0->cudf) (1.16.0)
Requirement already satisfied: fastrlock>=0.5 in /opt/conda/lib/python3.8/site-packages (from cupy-cuda115->cudf) (0.8)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/lib/python3.8/site-packages (from packaging->cudf) (3.0.8)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment inst
ead: https://pip.pypa.io/warnings/venv
```

Loading dataset on GPU

We make use of cudf to load data onto the GPU. cudf is an alternative to Pandas and loads a dataframe directly onto the GPU.

[4]:
```
df = cudf.read_csv('heart_cleveland_upload.csv')
```

[5]:
```
df.head()
```

[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

[6]:
```
df.describe()
```

[6]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000
mean	54.542088	0.676768	2.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	0.602694	0.676768	0.835017	0.461279
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187	0.938965	0.956690	0.499340
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	2.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	56.000000	1.000000	2.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	0.000000	0.000000
75%	61.000000	1.000000	3.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	1.000000	1.000000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	3.000000	2.000000	1.000000

Checking Missing Values

[7]:
```
#missing values
df.isna().sum()
```

[7]:
```
age          0
sex          0
cp           0
trestbps     0
chol         0
fbs          0
restecg      0
thalach      0
exang        0
oldpeak      0
slope        0
ca           0
thal         0
condition    0
dtype: int64
```

Number of classes

We have 2 classes in the dataset:

- 0 = No heart disease
- 1 = Heart disease

[9]:
```
df['condition'].nunique()
```

[9]: 2

Categorical variables

```
[10]: print(df['sex'].nunique())
2

[11]: print(df['sex'].unique())
0    0
1    1
Name: sex, dtype: int64

[12]: print(df['exang'].nunique())
2

[13]: print(df['exang'].unique())
0    0
1    1
Name: exang, dtype: int64
```

The performance of various models on the heart disease dataset is shown in thefollowing table.

Model	**Accuracy**
Chen, Austin H., et al.[2]	80%
Darmawahyuni et al. [3]	**96%**
Sharma et al. [4]	90.78%

References

1. UCI Machine Learning Repository [homepage on the Internet]. Arlington: The Association; 2006 [updated 1996 Dec 3; cited 2011 Feb 2]. Available from: http://archive.ics.uci.edu/ml/datasets/Heart+Disease
2. Chen, Austin H., et al. "HDPS: Heart disease prediction system." *2011 computingin Cardiology*. IEEE, 2011.
3. Darmawahyuni, Annisa, Siti Nurmaini, and Firdaus Firdaus. "Coronary heart disease interpretation based on deep neural network." *Computer Engineeringand Applications Journal* 8.1 (2019): 1-12.
4. Sharma, Sumit, and Mahesh Parmar. "Heart diseases prediction using deep learning neural network model." *International Journal of Innovative Technologyand Exploring Engineering (IJITEE)* 9.3 (2020): 2244-2248.

b) California Housing dataset

The California Housing dataset [1] is a well-known dataset in the machine learning community that contains information on the median house value and various other features for various regions in California. This dataset was derived from the 1990 U.S.census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, thesecolumns may take surprisingly large values for block groups with few households
and many empty houses, such as vacation resorts. The target variable is the medianhouse value for California districts, expressed in hundreds of thousands of dollars ($100,000).

The dataset contains approximately 20,000 instances and 10 attributes. Theseattributes are described as follows:

- longitude
- latitude
- housing_median_age
- total_rooms
- total_bedrooms
- population
- households
- median_income
- median_house_value
- ocean_proximity

This dataset has been widely used in machine learning research, particularly in the areaof regression modeling. The goal of most studies using this dataset is to develop models that can accurately predict the median house value of a given region based on the various features. It provides a convenient and accessible way to learn about the basics of machine learning, and it provides a useful benchmark for evaluating the performance of different algorithms. As machine learning continues to grow and evolve,this dataset will likely continue to play an important role in advancing our understanding of the real estate market and the use of machine learning in the field of real estate.

[1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/california-housing-prices/housing.csv
```

[2]:
```
!pip install cudf
```

```
Requirement already satisfied: cudf in /opt/conda/lib/python3.7/site-packages (21.10.1)
Requirement already satisfied: numba>=0.53.1 in /opt/conda/lib/python3.7/site-packages (from cudf) (0.55.2)
Requirement already satisfied: Cython<0.30,>=0.29 in /opt/conda/lib/python3.7/site-packages (from cudf) (0.29.33)
Requirement already satisfied: fastavro>=0.22.9 in /opt/conda/lib/python3.7/site-packages (from cudf) (1.6.1)
Requirement already satisfied: fsspec>=0.6.0 in /opt/conda/lib/python3.7/site-packages (from cudf) (2023.1.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from cudf) (1.21.6)
Requirement already satisfied: pandas<1.4.0dev0,>=1.0 in /opt/conda/lib/python3.7/site-packages (from cudf) (1.3.5)
Requirement already satisfied: typing_extensions in /opt/conda/lib/python3.7/site-packages (from cudf) (4.1.1)
Requirement already satisfied: protobuf in /opt/conda/lib/python3.7/site-packages (from cudf) (3.20.3)
Requirement already satisfied: nvtx>=0.2.1 in /opt/conda/lib/python3.7/site-packages (from cudf) (0.2.3)
Requirement already satisfied: cachetools in /opt/conda/lib/python3.7/site-packages (from cudf) (4.2.4)
Requirement already satisfied: packaging in /opt/conda/lib/python3.7/site-packages (from cudf) (23.0)
Requirement already satisfied: cupy-cuda110 in /opt/conda/lib/python3.7/site-packages (from cudf) (11.5.0)
Requirement already satisfied: llvmlite<0.39,>=0.38.0rc1 in /opt/conda/lib/python3.7/site-packages (from numba>=0.53.1->cudf) (0.38.1)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages (from numba>=0.53.1->cudf) (59.8.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas<1.4.0dev0,>=1.0->cudf) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas<1.4.0dev0,>=1.0->cudf) (2022.1)
Requirement already satisfied: fastrlock>=0.5 in /opt/conda/lib/python3.7/site-packages (from cupy-cuda110->cudf) (0.8)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.7.3->pandas<1.4.0dev0,>=1.0->cudf) (1.15.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment inst
ead: https://pip.pypa.io/warnings/venv class="ansi-yellow-fg">
```

[3]:
```
import cudf
```

Loading dataset onto the GPU

6]:
```
df = cudf.read_csv('housing.csv')
```

7]:
```
df.head()
```

7]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

8]:
```
df.describe()
```

8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

Checking null values

[9]:
```
#checking null values
df.isnull().sum()
```

[9]:
```
longitude               0
latitude                0
housing_median_age      0
total_rooms             0
total_bedrooms        207
population              0
households              0
median_income           0
median_house_value      0
ocean_proximity         0
dtype: int64
```

There are 207 rows with missing values for total_bedrooms field

[10]:
```
rows = df[df['total_bedrooms'].isnull()]
```

[11]:
```
rows.head()
```

[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
290	-122.16	37.77	47.0	1256.0	<NA>	570.0	218.0	4.3750	161900.0	NEAR BAY
341	-122.17	37.75	38.0	992.0	<NA>	732.0	259.0	1.6196	85100.0	NEAR BAY
538	-122.28	37.78	29.0	5154.0	<NA>	3741.0	1273.0	2.5762	173400.0	NEAR BAY
563	-122.24	37.75	45.0	891.0	<NA>	384.0	146.0	4.9489	247100.0	NEAR BAY
696	-122.10	37.69	41.0	746.0	<NA>	387.0	161.0	3.9063	178400.0	NEAR BAY

Checking number of categorical values

```
[12]: df['ocean_proximity'].nunique()

[12]: 5

[13]: df['ocean_proximity'].unique()

[13]: 0     <1H OCEAN
      1        INLAND
      2        ISLAND
      3      NEAR BAY
      4    NEAR OCEAN
      Name: ocean_proximity, dtype: object
```

The performance of various models on the California Housing dataset is shown in the following table.

Model	**Root Mean Squared Error**	**Mean Squared Error**
Xueheng et al. [2]	0.1508	-
Buyang et al. [3]	-	41811.422310
Guang-Bin et al. [4]	0.1365	-

References

1. Pace, R. Kelley, and Ronald Barry. "Sparse spatial autoregressions." *Statistics & Probability Letters* 33.3 (1997): 291-297.
2. Qiu, Xueheng, et al. "Ensemble deep learning for regression and time series forecasting." *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*. IEEE, 2014.
3. Cao, Buyang, and Bowen Yang. "Research on ensemble learning-based housingprice prediction model." *Big Geospatial Data and Data Science* 1.1 (2018): 1-8.
4. Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: a new learning scheme of feedforward neural networks." *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*. Vol. 2. Ieee, 2004.

2. Study two tools used for web scraping and write a description of each. In addition, implement web scraping using one of the tools.

Web scraping is a technique of extracting information from websites. It involves makingHTTP requests to a website's server, downloading the HTML content of the web page, and then parsing that data to extract the relevant information. Web scraping can be used to gather a large amount of data in a short amount of time and can help automatetedious manual data collection tasks. 2 tools used for web scraping are described below:

Beautiful Soup

Beautiful Soup is a Python library that is used for web scraping purposes to pull the dataout of HTML and XML files. It was created to make it easier to extract data from websites, without having to manually examine the HTML code of the site. It is designed to be a useful tool for web developers and data scientists, providing a way to extract data from websites quickly and easily. The library provides a number of different functions that can be used to parse and extract information from web pages.

One of the main advantages of Beautiful Soup is its ability to handle incomplete or broken HTML code. This is a common problem when scraping websites, as many pages may have malformed HTML code that makes it difficult to extract information.Beautiful Soup is designed to work around these issues, providing a way to parse through the HTML code and extract only the data that is needed.

Another advantage of Beautiful Soup is its ability to handle multiple different types ofdata formats. In addition to HTML and XML, the library can also handle other data formats, such as CSV and JSON, making it a versatile tool for data extraction.

Beautiful Soup works by first parsing the HTML or XML code of a website into a data structure, called a parse tree. This parse tree is then used to extract the data that is needed. The library provides a number of different functions that can be used to searchthe parse tree for specific data, such as tags, attributes, and text.

Scrapy

Scrapy is an open-source and collaborative web crawling framework for Python. It is used to extract the data from websites and is designed to be fast, efficient, and easy touse. Scrapy provides a number of built-in tools for managing and organizing web crawls, making it a popular choice for data extraction and web scraping tasks. It can follow links, extract data, and store the data in a structured format, such as CSV or JSON. Scrapy also provides tools for handling common web scraping challenges, suchas handling pagination and dealing with broken links.

A main advantage of Scrapy is its ability to handle large amounts of data. It is designedto be highly scalable, enabling it to handle large web scraping tasks without having to worry about performance issues.

In conclusion, Scrapy is a powerful and flexible web scraping framework that provides anumber of tools for managing and organizing web crawls.

Scraping articles using Beautiful Soup

We make use of Beautiful Soup library to scrape data science articles from a bloggingwebsite, namely Medium. The scraped articles are stored in a csv file namely scraped_content.csv

```
[10] import requests
     from bs4 import BeautifulSoup
     import pandas as pd
     import random

     url = 'https://towardsdatascience.com/archive/2023'
     res = requests.get(url)

[12] parsedHtml = BeautifulSoup(res.text, 'html.parser')

[13] stories = parsedHtml.find_all('div', class_='streamItem streamItem--postPreview js-streamItem')

formatted_stories = []
for story in stories:
    # get the title of the story
    story_title = story.find('h3').text if story.find('h3') else 'N/A'
    # get the subtitle of the story
    story_subtitle = story.find('h4').text if story.find('h4') else 'N/A'

    # get the number of claps
    clap_button = story.find('button', class_='button button--chromeless u-baseColor--buttonNormal js-multirecommendCountButton u-disablePointerEvents')
    claps = 0
    if (clap_button):
        # if clap button has a DOM reference, obtain its text
        claps = clap_button.text

    # get reference to the card header containing author info
    author_header = story.find('div', class_='postMetaInline u-floatLeft u-sm-maxWidthFullWidth')
    # access the reading time span element and get its title attribute
    reading_time = author_header.find('span', class_='readingTime')['title']

    # get read more ref
    read_more_ref = story.find('a', class_='button button--smaller button--chromeless u-baseColor--buttonNormal')
    url = read_more_ref['href'] if read_more_ref else 'N/A'

    formatted_stories.append([story_title, story_subtitle, claps, reading_time, url])

[15] medium_df = pd.DataFrame(formatted_stories, columns=['title', 'subtitle', 'claps', 'reading_time', 'article_link'])

[16] medium_df.to_csv('scraped_data.csv', index=False)
```

title	subtitle	claps	reading_time	article_link
How ChatGPT Works: The Model Behind The Bot	A brief introduction to the intuition and methodology…	2.9K	8 min read	https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286?source=collection_archive---------0-----------------------
Can ChatGPT Write Better SQL than a Data Analyst?	N/A	1.93K	6 min read	https://towardsdatascience.com/can-chatgpt-write-better-sql-than-a-data-analyst-f079518efab2?source=collection_archive---------1-----------------------
5 Python Tricks That Distinguish Senior Developers From Juniors	Illustrated through differences in…	409	6 min read	https://towardsdatascience.com/5-python-tricks-that-distinguish-senior-developers-from-juniors-826d57ab3940?source=collection_archive---------2-----------------------
12 Python Decorators to Take Your Code to the Next Level	Boost Your Programming Skills	502	11 min read	https://towardsdatascience.com/12-python-decorators-to-take-your-code-to-the-next-level-a910a1ab3e99?source=collection_archive---------3-----------------------
7 of the Most Used Feature Engineering Techniques	Hands-on Feature Engineering with Scikit-Learn…	535	37 min read	https://towardsdatascience.com/7-of-the-most-used-feature-engineering-techniques-bcc50f48474d?source=collection_archive---------4-----------------------
4 Common Python Mistakes You Should Avoid as a Beginner	And how to correct yourself before you…	415	7 min read	https://towardsdatascience.com/4-common-python-mistakes-you-should-avoid-as-a-beginner-bd28feb6162b?source=collection_archive---------5-----------------------
How to Find the Best Theoretical Distribution for Your Data.	Knowing the underlying data distribution…	476	19 min read	https://towardsdatascience.com/how-to-find-the-best-theoretical-distribution-for-your-data-a26e5673b4bd?source=collection_archive---------6-----------------------
Pandas vs. Polars: A Syntax and Speed Comparison	Understanding the major differences between the…	280	7 min read	https://towardsdatascience.com/pandas-vs-polars-a-syntax-and-speed-comparison-5aa54e27497e?source=collection_archive---------7-----------------------
Speed Up your Python Skills in 2023	Seven tips to take you to the next level	701	8 min read	https://towardsdatascience.com/speed-up-your-python-skills-in-2023-e680f4c56f37?source=collection_archive---------8-----------------------
The Future of the Modern Data Stack in 2023	Featuring 4 new emerging trends and 6 big trends from last…	313	19 min read	https://towardsdatascience.com/the-future-of-the-modern-data-stack-in-2023-b08c2aed04e2?source=collection_archive---------9-----------------------

Practical 3

Create a project charter for any application.

Project Title: Launching an Online Clothing Store

Project Purpose:
The purpose of this project is to launch an online clothing store that offers high-quality and fashionable clothing for men and women. The store will be a one-stop-shop for customers who are looking for trendy and stylish clothing at affordable prices.

Project Scope:
The scope of this project includes the development of an online store, designing the website and logo, selecting and sourcing high-quality products, setting up an inventory management system, establishing partnerships with suppliers, and creating an effective marketing strategy.

Objectives:

1. Develop a user-friendly and attractive website that provides an easy shopping experience for customers.
2. Source high-quality and fashionable clothing for men and women that meets the customer's expectations and preferences.
3. Set up an effective inventory management system to ensure timely delivery of orders.
4. Establish partnerships with reputable suppliers to ensure the quality of the products and timely delivery.
5. Develop an effective marketing strategy to promote the store and increase its visibility and sales.

Deliverables:

1. A fully functional and user-friendly website.
2. A comprehensive inventory management system.
3. A selection of high-quality and fashionable clothing for men and women.
4. A list of reputable suppliers and established partnerships.
5. A comprehensive marketing plan that includes social media marketing, email marketing, and search engine optimization.

Timeline:

1. Website design and development - 2 months
2. Product sourcing and inventory management system setup - 2 months
3. Supplier partnerships established - 3 months
4. Marketing plan developed and implemented - 6 months
5. Launch of the online store - 6 months

Budget:
The total budget for this project is $200,000. This includes website design and development, product sourcing, inventory management system setup, supplier partnerships, and marketing expenses.

Project Manager:

John Doe will be the project manager for this project. He will be responsible for overseeing the project, ensuring that it stays within budget and on schedule, and ensuring that all project objectives are met.

Stakeholders:

1. Project Sponsor: Jane Smith
2. Project Manager: John Doe
3. IT Team
4. Marketing Team
5. Suppliers
6. Customers

Assumptions:

1. The IT team has the necessary skills and resources to design and develop the website.
2. The suppliers can provide high-quality products at reasonable prices.
3. The marketing team can effectively promote the store and attract customers.
4. The project will stay within budget and on schedule.
5. The online store will be well received by customers and generate significant revenue.

Risks:

1. Technical issues may arise during the website development process.
2. Sourcing high-quality products at reasonable prices may be a challenge.
3. Delay in the delivery of products from suppliers may affect the inventory management system.
4. Marketing campaigns may not be successful in attracting customers.
5. Competition from other online clothing stores may affect the sales of the store.

Practical 4

Implement any Five Graphs you studied and derive knowledge using them.

In [1]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import matplotlib.pyplot as plt
import seaborn as sns
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/iris/Iris.csv
/kaggle/input/iris/database.sqlite
```

In [3]:

```
df = pd.read_csv('/kaggle/input/iris/Iris.csv')
df = df.drop('Id',axis=1)
df.head()
```

Out[3]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:

```
df.describe()
```

Out[4]:

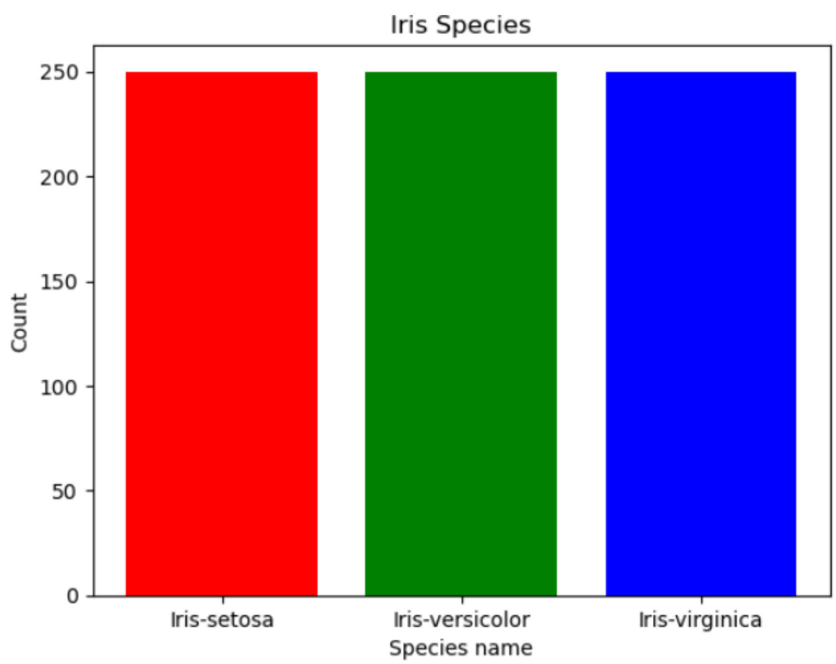
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Bar Graph

We analyze the distribution of data in the different classes

In [7]:

```
species = [df[df['Species'] == 'Iris-setosa'].size,df[df['Species'] == 'Iris-versicolor'].size,df[df
['Species'] == 'Iris-virginica'].size ]
plt.bar('Iris-setosa',species[0],color = 'red')
plt.bar('Iris-versicolor',species[1],color = 'green')
plt.bar('Iris-virginica',species[2],color = 'blue')
plt.title('Iris Species')
plt.xlabel('Species name')
plt.ylabel('Count')
plt.show()
```



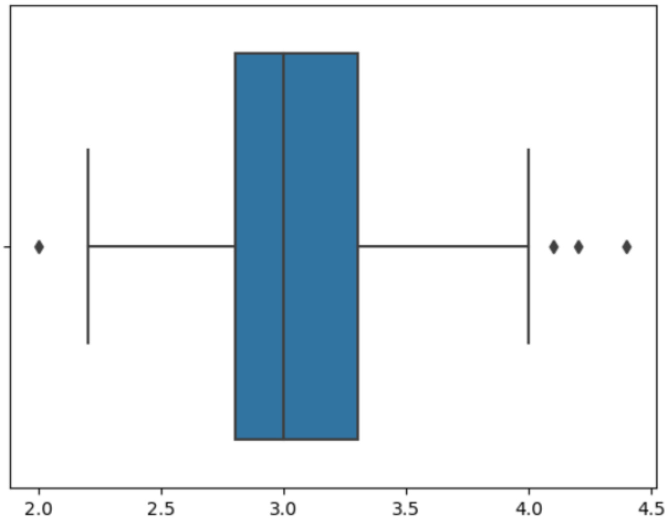
We see that there is an equal distribution of data in each of the classes

Boxplot

We analyze the outliers in Sepal Width and Sepal Length using Boxplot

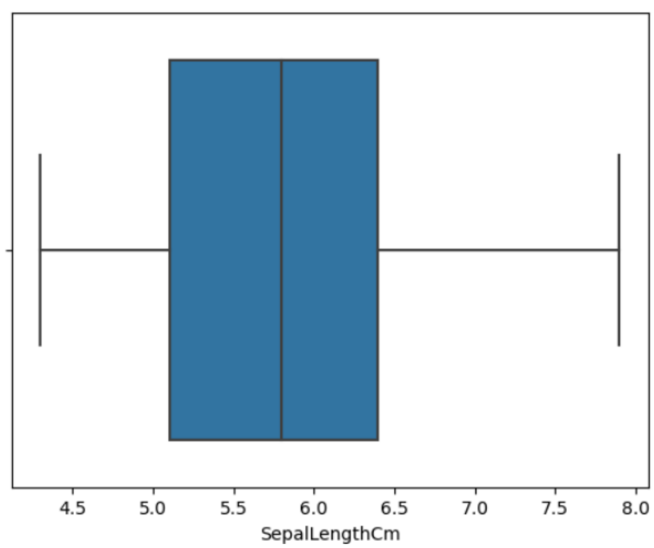
In [12]:

```
sns.boxplot(df,x='SepalWidthCm')
plt.show()
```



In [13]:

```
sns.boxplot(df,x='SepalLengthCm')
plt.show()
```



Scatter Plot

We analyze the relation between Petal Length and Petal Width using Scatter Plot

In [14]:
```
plt.scatter(df['PetalLengthCm'].values,df['PetalWidthCm'].values)
plt.title("Scatter plot of Petal Length and Petal Width")
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
```

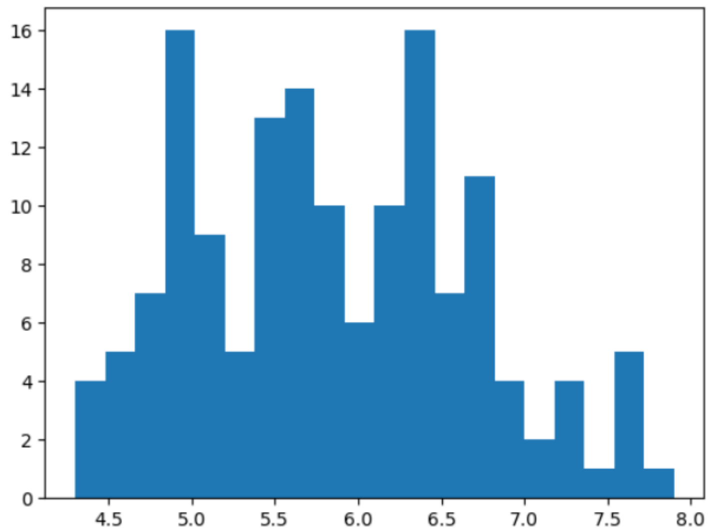
The relation between Petal Length and Petal Width is linear and positive

Histogram

We analyze the distribution of Sepal Length values using histogram

In [15]:

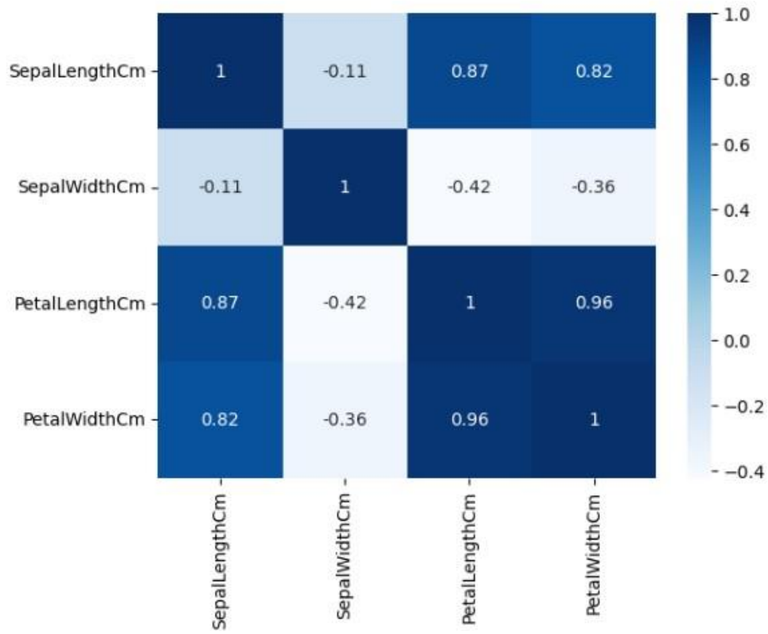
```
plt.hist(df['SepalLengthCm'].values,bins=20)
plt.show()
```



Heatmap

In [18]:

```
mat = df.corr()
sns.heatmap(mat,cmap='Blues',annot=True)
plt.show()
```



Practical 5

Implement PCA for classification using CUDA toolkit developed by NVIDIA.

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import torch
```

In [2]:
```
device = torch.device('cuda')
```

In [3]:
```
import torch
import tqdm as notebook_tqdm
import pandas as pd
from sklearn.preprocessing import StandardScaler
from torch.linalg import eig
```

In [4]:
```
df = pd.read_csv('/kaggle/input/breast-cancer-wisconsin-data/data.csv')
```

In [5]:
```
df = df.drop(['id','Unnamed: 32'],axis=1)
```

In [6]:
```
df.head()
```

Out[6]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_w
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	25.38
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	24.99
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	23.57
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	14.91
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	22.54

5 rows × 31 columns

In [9]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['diagnosis'] = le.fit_transform(df['diagnosis'])
df.head()
```

Out[9]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_w
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	...	25.38
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	...	24.99
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	...	23.57
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	...	14.91
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	...	22.54

In [10]:

```
y = df['diagnosis']
y.head()
```

Out[10]:

```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

In [11]:

```
X = df.drop('diagnosis',axis=1)
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [12]:

```
X_tensor = torch.tensor(X,dtype = torch.float32)
y_tensor = torch.tensor(y)
```

In [13]:

```
print(X_tensor.device)
print(y_tensor.device)
```

```
cpu
cpu
```

In [14]:

```
X_tensor = X_tensor.to(device)
y_tensor = y_tensor.to(device)
```

In [15]:

```
print(X_tensor.device)
print(y_tensor.device)
```

```
cuda:0
cuda:0
```

In [17]:

```
pca = torch.pca_lowrank(X_tensor)
```

In [23]:

```
print(pca)
```

```
(tensor([[-0.1058,  0.0343, -0.0295,  0.1070, -0.0366, -0.0780],
        [-0.0275, -0.0662, -0.0137,  0.0319,  0.0204, -0.0012],
        [-0.0660, -0.0189, -0.0132,  0.0301,  0.0001, -0.0100],
        ...,
        [-0.0145, -0.0334,  0.0145, -0.0653,  0.0560,  0.0296],
        [-0.1193,  0.0294, -0.0464, -0.0699, -0.0044, -0.0310],
        [ 0.0630, -0.0117,  0.0386, -0.0666, -0.0069, -0.0687]],
       device='cuda:0'), tensor([86.9323, 56.9067, 40.0190, 33.4112, 30.4548, 25.1822], device='cuda:0'), tensor([[-2.1882e-01, -
2.3415e-01, -9.5189e-03,  4.7972e-02,  4.2788e-02,
         -9.9945e-03],
        [-1.0375e-01, -5.9459e-02,  7.7054e-02, -5.9277e-01, -6.3556e-02,
          5.5827e-02],
```

In [25]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [26]:

```
print(y.shape)
```

```
(569,)
```

In [27]:

```
print(X_pca)
```

```
tensor([[-0.1058,  0.0343, -0.0295,  0.1070, -0.0366],
        [-0.0275, -0.0662, -0.0137,  0.0319,  0.0204],
        [-0.0660, -0.0189, -0.0132,  0.0301,  0.0001],
        ...,
        [-0.0145, -0.0334,  0.0145, -0.0653,  0.0560],
        [-0.1193,  0.0294, -0.0464, -0.0699, -0.0044],
        [ 0.0630, -0.0117,  0.0386, -0.0666, -0.0069]], device='cuda:0')
```

In [28]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_pca,y,test_size=0.20)
```

In [51]:

```
y_pred = []
y_train_lst = y_train.tolist()
```

In [52]:

```
for test in X_test:
    distances = []
    for idx,train in enumerate(X_train):
        dist = torch.norm(train - test)
        distances.append((dist,idx))
    distances.sort(key=lambda x:x[0])
    y_pred.append(y_train_lst[distances[0][1]])
```

In [53]:

```
cf_matrix = confusion_matrix(y_test, y_pred)
cf_matrix
```

Out[53]:

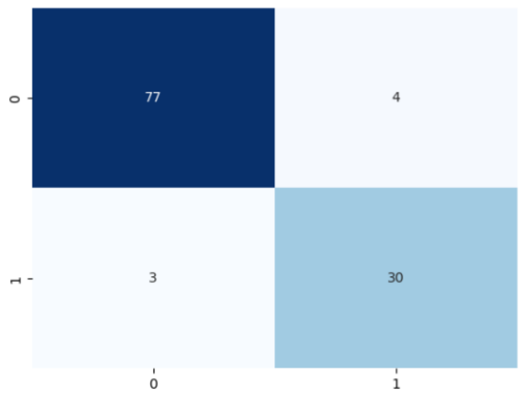
```
array([[77,  4],
       [ 3, 30]])
```

In [54]:

```
import seaborn as sns
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test,y_pred)
sns.heatmap(cf_matrix,cmap='Blues',annot=True,fmt='g',cbar=False)
plt.show()
print(acc)
```

In [54]:

```
import seaborn as sns
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test,y_pred)
sns.heatmap(cf_matrix,cmap='Blues',annot=True,fmt='g',cbar=False)
plt.show()
print(acc)
```




```
0.9385964912280702
```

Practical 6

Implement supervised Feature Selection strategy (either forward/backward). Evaluate performance for intermediate selected feature subsets.

In [31]:
```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/heart-disease-cleveland-uci/heart_cleveland_upload.csv
```

In [32]:
```
df = pd.read_csv('/kaggle/input/heart-disease-cleveland-uci/heart_cleveland_upload.csv')
df.head()
```

Out[32]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	condition
0	69	1	0	160	234	1	2	131	0	0.1	1	1	0	0
1	69	0	0	140	239	0	0	151	0	1.8	0	2	0	0
2	66	0	0	150	226	0	0	114	0	2.6	2	0	0	0
3	65	1	0	138	282	1	2	174	0	1.4	1	1	0	1
4	64	1	0	110	211	0	2	144	1	1.8	1	0	0	0

In [33]:
```
df.describe()
```

Out[33]:

Out[33]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.000000	297.00000
mean	54.542088	0.676768	2.158249	131.693603	247.350168	0.144781	0.996633	149.599327	0.326599	1.055556	0.602694	0.676768
std	9.049736	0.468500	0.964859	17.762806	51.997583	0.352474	0.994914	22.941562	0.469761	1.166123	0.618187	0.938965
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	48.000000	0.000000	2.000000	120.000000	211.000000	0.000000	0.000000	133.000000	0.000000	0.000000	0.000000	0.000000
50%	56.000000	1.000000	2.000000	130.000000	243.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	3.000000	140.000000	276.000000	0.000000	2.000000	166.000000	1.000000	1.600000	1.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	3.000000

In [34]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
```

In [35]:
```
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

In [36]:
```
df = df.drop('condition',axis=1)
```

In [37]:

```
X_train,X_test,y_train,y_test = train_test_split(df,y,test_size=0.25,random_state=1)
X_train,X_val,y_train,y_val = train_test_split(X_train,y_train,test_size=0.2,random_state=1)
```

In [38]:

```
def evaluate_metric(model, x_test, y_test):
    return accuracy_score(y_test, model.predict(x_test))
```

In [39]:

```
def forward_feature_selection(x_train, x_cv, y_train, y_cv, n):
    feature_set = []
    for num_features in range(n):
        metric_list = []
        model = KNeighborsClassifier(n_neighbors=3)
        for feature in x_train.columns:
            if feature not in feature_set:
                f_set = feature_set.copy()
                f_set.append(feature)
                model.fit(x_train[f_set], y_train)
                metric_list.append((evaluate_metric(model, x_cv[f_set], y_cv), feature))
        metric_list.sort(key=lambda x : x[0], reverse = True)
        feature_set.append(metric_list[0][1])
        print(f'The best accuracy in iteration {num_features+1} is {metric_list[0][0]} and the feature selected is {metric_list[0]
[1]}')
    return feature_set
```

In [40]:

```
f=forward_feature_selection(X_train, X_val, y_train, y_val, 5)
```

```
The best accuracy in iteration 1 is 0.7333333333333333 and the feature selected is ca
The best accuracy in iteration 2 is 0.8 and the feature selected is sex
The best accuracy in iteration 3 is 0.8444444444444444 and the feature selected is slope
The best accuracy in iteration 4 is 0.8666666666666667 and the feature selected is fbs
The best accuracy in iteration 5 is 0.8666666666666667 and the feature selected is thal
```

In [41]:

```
print(f)
```

```
['ca', 'sex', 'slope', 'fbs', 'thal']
```

In [42]:

```
X_train = X_train[f].values
X_test = X_test[f].values
```

In [43]:

```
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,y_train)
```

Out[43]:

```
KNeighborsClassifier(n_neighbors=3)
```

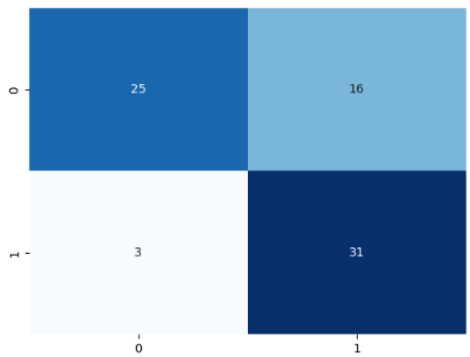
In [44]:

```
y_pred = model.predict(X_test)
acc = accuracy_score(y_test,y_pred)
print(acc)
```

```
0.7466666666666667
```

In [45]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt='g',cbar=False,cmap='Blues')
plt.show()
```



Practical 7

Implement Neural Network for any application using CUDA toolkit developed by NVIDIA.

```
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.datasets as datasets
import torchvision.transforms as transforms

# Set device to GPU if available, else use CPU
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)
# Define the neural network architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(784, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, 10)
        self.relu = nn.ReLU()
```

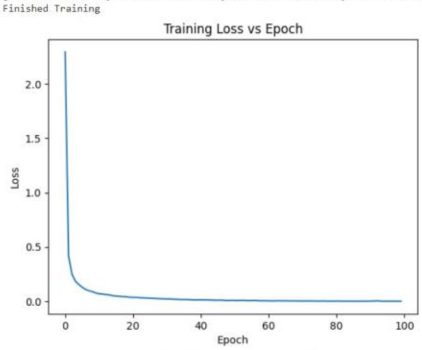
```
    def forward(self, x):
        x = x.view(-1, 784)
        x = self.relu(self.fc1(x))
        x = self.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# Define the loss function and optimizer
net = Net().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)

# Load the MNIST dataset
train_dataset = datasets.MNIST(root='./data', train=True, transform=transforms.ToTensor(), download=True)
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32, shuffle=True)
train_losses = []
n_epochs = 10
```

```
# Train the neural network
for epoch in range(n_epochs):
    running_loss = 0.0
    for i, (inputs, labels) in enumerate(train_loader, 0):
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        if i % 1000 == 999:    # print every 1000 mini-batches
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 1000))
            train_losses.append(running_loss/1000)
            running_loss = 0.0
print(train_losses)
print('Finished Training')
plt.plot(range(n_epochs), train_losses)
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss vs Epoch')
plt.show()
```

```python
# Test the neural network on a small subset of the MNIST test set
test_dataset = datasets.MNIST(root='./data', train=False, transform=transforms.ToTensor(), download=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=32, shuffle=True)
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = net(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / total))
```

```
cuda
[1,  1000] loss: 2.296
[2,  1000] loss: 0.417
[3,  1000] loss: 0.248
[4,  1000] loss: 0.186
[5,  1000] loss: 0.155
[6,  1000] loss: 0.129
[7,  1000] loss: 0.109
[8,  1000] loss: 0.097
[9,  1000] loss: 0.088
[10,  1000] loss: 0.076
[11,  1000] loss: 0.069
[12,  1000] loss: 0.066
[13,  1000] loss: 0.061
[14,  1000] loss: 0.058
[15,  1000] loss: 0.051
[16,  1000] loss: 0.047
[17,  1000] loss: 0.045
[18,  1000] loss: 0.042
[19,  1000] loss: 0.042
[20,  1000] loss: 0.037
[21,  1000] loss: 0.036
[22,  1000] loss: 0.036
[23,  1000] loss: 0.032
[24,  1000] loss: 0.031
[25,  1000] loss: 0.029
[26,  1000] loss: 0.028
[27,  1000] loss: 0.026
```

```
[96,  1000] loss: 0.001
[97,  1000] loss: 0.001
[98,  1000] loss: 0.000
[99,  1000] loss: 0.000
[100,  1000] loss: 0.000
[2.295739882469177, 0.4171782578751445, 0.24829997172765433, 0.18555129576288162, 0.15454984938539565, 0.12900534641509875, 0.10908874569786713, 0.09673536322917789, 0.08808159982331563, 0.07628849097259
Finished Training
```




```
Accuracy of the network on the 10000 test images: 98 %
```

Practical 8

Implement Bagging and Boosting concept of Ensemble Learning show performance enhance using Bagging and Boosting.

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

```
df = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/wine//wine.data', header=None)
df.columns = ['Class label', 'Alcohol', 'Malic acid', 'Ash',
              'Alcalinity of ash', 'Magnesium', 'Total phenols',
              'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
              'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']
```

```
df.head()
```

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735

```
X = df.iloc[:,1:].values
y = df.iloc[:,0].values
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

Evaluating the decision tree model

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

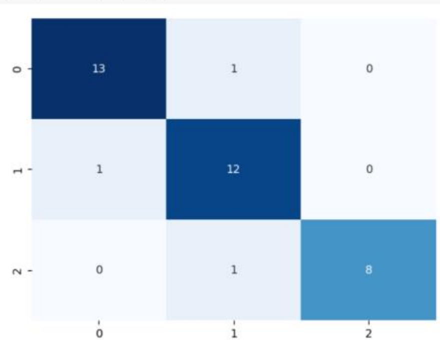
```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
model = DecisionTreeClassifier(max_depth=3,random_state=0)
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
y_pred = model.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm = confusion_matrix(y_test, y_pred)
acc = accuracy_score(y_test, y_pred)
```

```
sns.heatmap(cm, cmap='Blues', fmt='g', annot=True, cbar=False)
plt.show()
print(f'Accuracy: {acc}')
```




```
Accuracy: 0.9166666666666666
```

```
[ ] feature_names = df.columns.values[1:]
    print(feature_names)
```

```
['Alcohol' 'Malic acid' 'Ash' 'Alcalinity of ash' 'Magnesium'
 'Total phenols' 'Flavanoids' 'Nonflavanoid phenols' 'Proanthocyanins'
 'Color intensity' 'Hue' 'OD280/OD315 of diluted wines' 'Proline']
```

```
[ ] if not os.path.exists("/content/drive/MyDrive/output/") : os.mkdir("/content/drive/MyDrive/output/")
    export_graphviz(
        model,
        out_file='/content/drive/MyDrive/output/tree.dot',
        feature_names=feature_names
    )
```

```
[ ] !pip install graphviz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: graphviz in /usr/local/lib/python3.9/dist-packages (0.20.1)
```

```
[ ] !dot -Tpng /content/drive/MyDrive/output/tree.dot -o /content/drive/MyDrive/output/fig-tree.png
```

Evaluating Bagging Model

```
[ ] from sklearn.ensemble import BaggingClassifier
    tree = DecisionTreeClassifier()
    bagging_clf = BaggingClassifier(base_estimator=tree, n_estimators=1500, random_state=42)
    bagging_clf.fit(X_train, y_train)
```

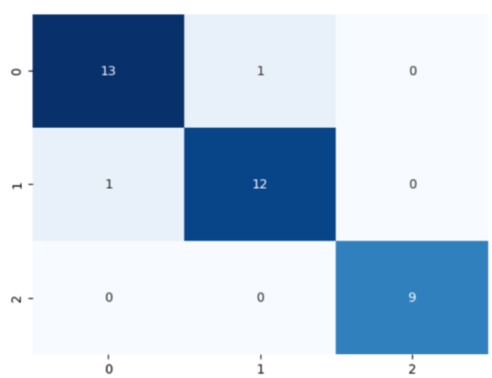
```
/usr/local/lib/python3.9/dist-packages/sklearn/ensemble/_base.py:166: FutureWarning: `base_estimator` was renamed to `estimator` in version 1.2 and will be removed in 1.
  warnings.warn(
```

```
BaggingClassifier
base_estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
[ ] y_pred_bag = bagging_clf.predict(X_test)
```

```
[ ] cm_bag = confusion_matrix(y_test, y_pred_bag)
    acc_bag = accuracy_score(y_test, y_pred_bag)
```

```
[ ] sns.heatmap(cm_bag, cmap='Blues', fmt='g', annot=True, cbar=False)
    plt.show()
    print(f'Accuracy: {acc_bag}')
```




```
Accuracy: 0.9444444444444444
```

Evaluating Boosting Model

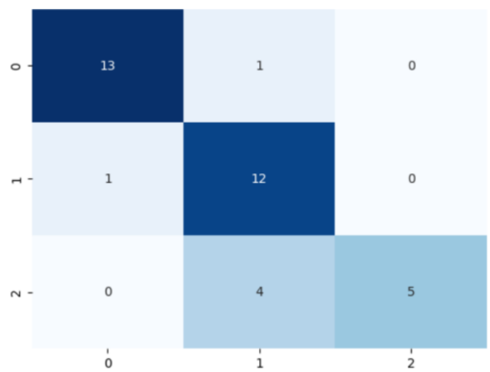
```
from sklearn.ensemble import AdaBoostClassifier
ada_boost_clf = AdaBoostClassifier(n_estimators=100)
ada_boost_clf.fit(X_train, y_train)
```

```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=100)
```

```
y_pred_boost = ada_boost_clf.predict(X_test)
```

```
cm_boost = confusion_matrix(y_test, y_pred_boost)
acc_boost = accuracy_score(y_test, y_pred_boost)
```

```
sns.heatmap(cm_boost, cmap='Blues', fmt='g', annot=True, cbar=False)
plt.show()
print(f'Accuracy: {acc_boost}')
```




```
Accuracy: 0.8333333333333334
```