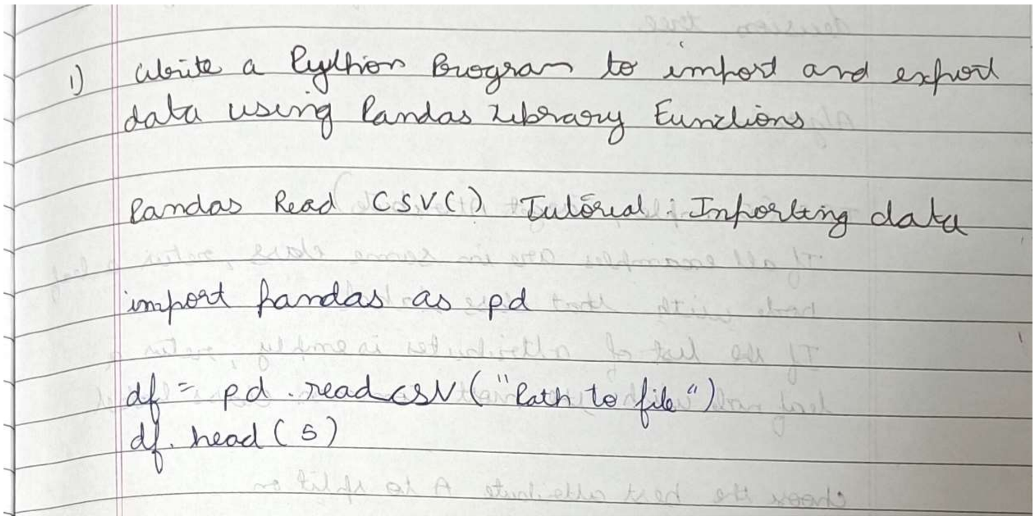
LAB 1

Write a python program to import and export data using Pandas library functions.

Importing data

Algorithm(Observation book)



1) Write a Python Program to import and export data using Pandas Library Functions

Pandas Read CSV() Tutorial : Importing data

import pandas as pd

df = pd.read_csv("Path to file")
df.head(5)

Code

```
import pandas as pd
df=pd.read_csv("/content/austinHousingData.csv")
df.head(5)
```

Output

	zpid	city	streetAddress	zipcode	description	latitude	longitude	propertyTaxRate	garageSpaces	hasAssociation	...
0	111373431	pflugerville	14424 Lake Victor Dr	78660	14424 Lake Victor Dr, Pflugerville, TX 78660 i...	30.430632	-97.663078	1.98	2	True	...
1	120900430	pflugerville	1104 Strickling Dr	78660	Absolutely GORGEOUS 4 Bedroom home with 2 full...	30.432673	-97.661697	1.98	2	True	...
2	2084491383	pflugerville	1408 Fort Dessau Rd	78660	Under construction - estimated completion in A...	30.409748	-97.639771	1.98	0	True	...
3	120901374	pflugerville	1025 Strickling Dr	78660	Absolutely darling one story home in charming ...	30.432112	-97.661659	1.98	2	True	...
4	60134862	pflugerville	15005 Donna Jane Loop	78660	Brimming with appeal & warm livability! Sleek ...	30.437368	-97.656860	1.98	0	True	...

5 rows × 47 columns

Exporting data

Algorithm(Observation book)

Exporting data

URL = "Iris data set link"
col_names = ["sepal_length_in_cm",
"sepal_width_in_cm",
"Petal_length_in_cm",
"Petal_width_in_cm",
"class"]

iris_data = pd.read_csv(url, names=col_names)
iris_data.head(5)

iris_data.to_csv("cleaned_iris_data.csv")

Code

```
url = “https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data”
# Define the column names
col_names = [“sepal_length_in_cm”,
            “sepal_width_in_cm”,
            “petal_length_in_cm”,
            “petal_width_in_cm”,
            “class”]
# Read data from URL
iris_data = pd.read_csv(url, names=col_names)
iris_data.head(5)
iris_data.to_csv(“/content/exported_irisData.csv”)
```

Output:

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
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

Demonstrate various data pre-processing techniques for a given dataset.

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
df1=pd.read_csv("/content/Data.csv")
df1.head(5)
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
#Identifying and handling the missing values
df1.isnull().sum()
```

```
Country      0
Age          1
Salary       1
Purchased    0
dtype: int64
```

LAB 2

Use an appropriate dataset for building the decision tree(ID3) and apply this knowledge to classify a new sample.

Algorithm(Observation book)

ML-Lab 2

Use an appropriate data set for building the decision tree

Algorithm

ID3 (Examples, Target_Attribute)
If all examples are in same class, return a leaf node with that class label
If the list of attributes is empty, return a leaf node with the most common class label

Choose the best attribute A to split on
Create a decision tree node with Attribute A

For each possible value v of A:
Add a new branch below the decision node for value v
Let Examples v be the subst of Examples with value v for attribute A

if Examples v is empty:
Add a leaf node with most common class label in examples to this branch
Else:
Recursively call ID3 (Examples_v, Target_Attribute) and add the returned subtree to this branch

Return the decision tree

Code

```
# Importing the required libraries
import pandas as pd
import numpy as np
import math
data = pd.read_csv('/content/PlayTennis.csv')
```

```
def highlight(cell_value):
    '''
    Highlight yes / no values in the dataframe
    '''
    color_1 = 'background-color: pink;'
    color_2 = 'background-color: lightgreen;'

    if cell_value == 'no':
        return color_1
    elif cell_value == 'yes':
        return color_2

data.style.applymap(highlight)\
    .set_properties(subset=data.columns, **{'width': '100px'})\
    .set_table_styles([{'selector': 'th', 'props': [('background-
color', 'lightgray'), ('border', '1px solid gray'),
                                                     ('font-weight',
'bold')]},
     {'selector': 'tr:hover', 'props': [('background-color', 'white'),
('border', '1.5px solid black')]}])
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
2	overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
6	overcast	cool	normal	True	yes
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
9	rainy	mild	normal	False	yes
10	sunny	mild	normal	True	yes
11	overcast	mild	high	True	yes
12	overcast	hot	normal	False	yes
13	rainy	mild	high	True	no

```python
def find_entropy(data):
    """
    Returns the entropy of the class or features
    formula: - ∑ P(X)logP(X)
    """
    entropy = 0
    for i in range(data.nunique()):
        x = data.value_counts()[i]/data.shape[0]
        entropy += (- x * math.log(x,2))
    return round(entropy,3)

def information_gain(data, data_):
    """
    Returns the information gain of the features
    """
    info = 0
    for i in range(data_.nunique()):
        df = data[data_ == data_.unique()[i]]
        w_avg = df.shape[0]/data.shape[0]
        entropy = find_entropy(df.play)
        x = w_avg * entropy
        info += x
    ig = find_entropy(data.play) - info
    return round(ig, 3)

def entropy_and_infogain(datax, feature):
    """
    Grouping features with the same class and computing their
    entropy and information gain for splitting
    """
    for i in range(data[feature].nunique()):
        df = datax[datax[feature]==data[feature].unique()[i]]
        if df.shape[0] < 1:
            continue

        display(df[[feature, 'play']].style.applymap(highlight)\
                .set_properties(subset=[feature, 'play'], **{'width':
'80px'})\
                .set_table_styles([{'selector': 'th', 'props':
[('background-color', 'lightgray'),
                                                                 ('borde
r', '1px solid gray'),
                                                                 ('font-
weight', 'bold')]},
                                   {'selector': 'td', 'props':
[('border', '1px solid gray')]},
                                   {'selector': 'tr:hover', 'props':
[('background-color', 'white'),
```

```
                                                                           (
'border', '1.5px solid black')]}]))

        print(f'Entropy of {feature} - {data[feature].unique()[i]} =
{find_entropy(df.play)}')
    print(f'Information Gain for {feature} = {information_gain(datax,
datax[feature])}')
```

```
#Computing entropy for the entire dataset
print(f'Entropy of the entire dataset: {find_entropy(data.play)}')
```

```
Entropy of the entire dataset: 0.94
```

```
#Calculate the Information Gain for each feature.
#Outlook
entropy_and_infogain(data, 'outlook')
```

	outlook	play
0	sunny	no
1	sunny	no
7	sunny	no
8	sunny	yes
10	sunny	yes

```
Entropy of outlook - sunny = 0.971
```

	outlook	play
2	overcast	yes
6	overcast	yes
11	overcast	yes
12	overcast	yes

```
Entropy of outlook - overcast = 0.0
```

	outlook	play
3	rainy	yes
4	rainy	yes
5	rainy	no
9	rainy	yes
13	rainy	no

```
Entropy of outlook - rainy = 0.971
Information Gain for outlook = 0.246
```

```python
#Temp
entropy_and_infogain(data, 'temp')
```

	temp	play
0	hot	no
1	hot	no
2	hot	yes
12	hot	yes

```
Entropy of temp - hot = 1.0
```

	temp	play
3	mild	yes
7	mild	no
9	mild	yes
10	mild	yes
11	mild	yes
13	mild	no

```
Entropy of temp - mild = 0.918
```

	temp	play
4	cool	yes
5	cool	no
6	cool	yes
8	cool	yes

```
Entropy of temp - cool = 0.811
Information Gain for temp = 0.029
```

```python
#Humidity
entropy_and_infogain(data, 'humidity')
```

	humidity	play
0	high	no
1	high	no
2	high	yes
3	high	yes
7	high	no
11	high	yes
13	high	no

```
Entropy of humidity - high = 0.985
```

	humidity	play
4	normal	yes
5	normal	no
6	normal	yes
8	normal	yes
9	normal	yes
10	normal	yes
12	normal	yes

```
Entropy of humidity - normal = 0.592
Information Gain for humidity = 0.151
```

```
#Windy
entropy_and_infogain(data, 'windy')
```

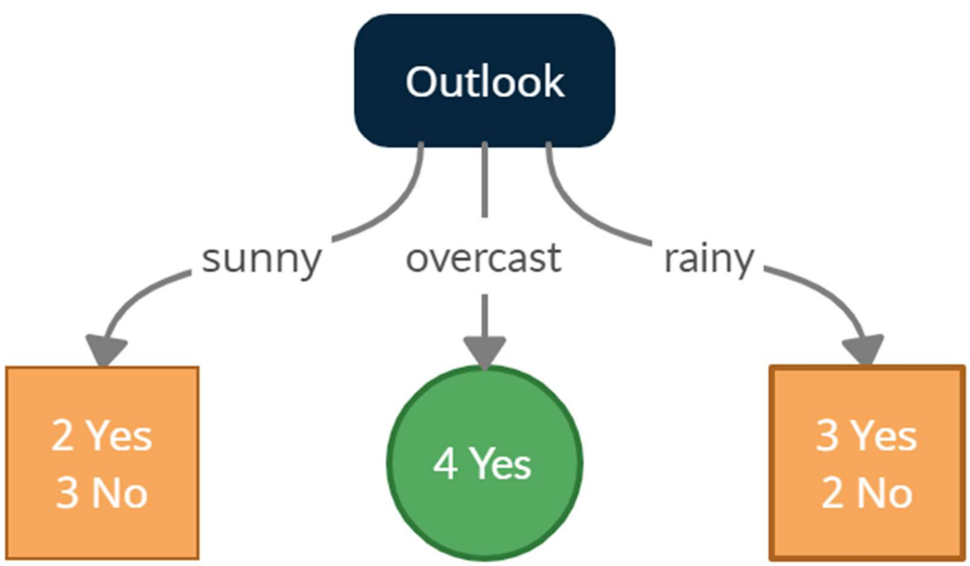
	windy	play
0	False	no
2	False	yes
3	False	yes
4	False	yes
7	False	no
8	False	yes
9	False	yes
12	False	yes

```
Entropy of windy - False = 0.811
```

	windy	play
1	True	no
5	True	no
6	True	yes
10	True	yes
11	True	yes
13	True	no

```
Entropy of windy - True = 1.0
Information Gain for windy = 0.048
```

#Make a decision tree node using the feature with the maximum Information Gain.



```
sunny = data[data['outlook'] == 'sunny']
sunny.style.applymap(highlight)\
    .set_properties(subset=data.columns, **{'width': '100px'})\
    .set_table_styles([{'selector': 'th', 'props': [('background-color', 'lightgray'), ('border', '1px solid gray'),
                                                    ('font-weight', 'bold')]},
     {'selector': 'tr:hover', 'props': [('background-color', 'white'), ('border', '1.5px solid black')]}])
```

	outlook	temp	humidity	windy	play
0	sunny	hot	high	False	no
1	sunny	hot	high	True	no
7	sunny	mild	high	False	no
8	sunny	cool	normal	False	yes
10	sunny	mild	normal	True	yes

```
print(f'Entropy of the Sunny dataset: {find_entropy(sunny.play)}')
```

```
Entropy of the Sunny dataset: 0.971
```

```
#temp
entropy_and_infogain(sunny, 'temp')
```

	temp	play
0	hot	no
1	hot	no

```
Entropy of temp - hot = 0.0
```

	temp	play
7	mild	no
10	mild	yes

```
Entropy of temp - mild = 1.0
```

	temp	play
8	cool	yes

```
Entropy of temp - cool = 0.0
Information Gain for temp = 0.571
```

```python
#Humidity
entropy_and_infogain(sunny, 'humidity')
```

	humidity	play
0	high	no
1	high	no
7	high	no

```
Entropy of humidity - high = 0.0
```

	humidity	play
8	normal	yes
10	normal	yes

```
Entropy of humidity - normal = 0.0
Information Gain for humidity = 0.971
```

```python
#Windy
entropy_and_infogain(sunny, 'windy')
```

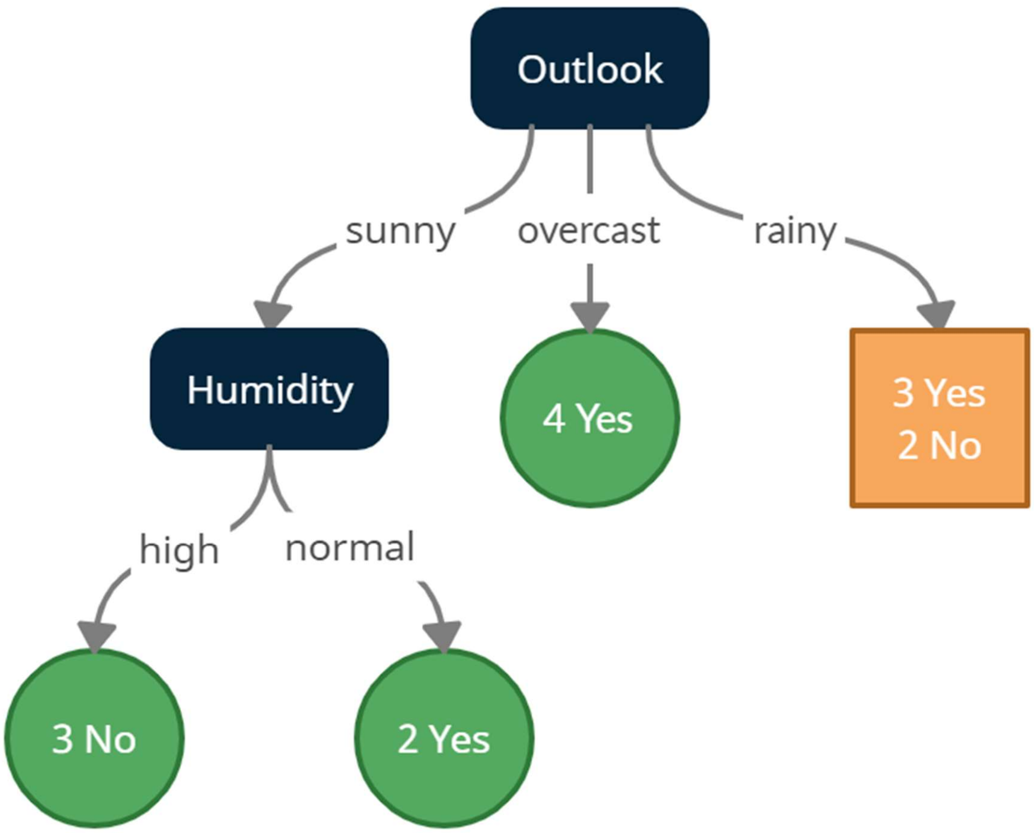
	windy	play
0	False	no
7	False	no
8	False	yes

```
Entropy of windy - False = 0.918
```

	windy	play
1	True	no
10	True	yes

```
Entropy of windy - True = 1.0
Information Gain for windy = 0.02
```

#Making a decision tree node using the feature which has the maximum Information Gain




```
#Outlook - Rainy
rainy = data[data['outlook'] == 'rainy']
rainy.style.applymap(highlight)\
    .set_properties(subset=data.columns, **{'width': '100px'})\
    .set_table_styles([{'selector': 'th', 'props': [('background-
color', 'lightgray'), ('border', '1px solid gray'),
                                                     ('font-weight',
'bold')]},
     {'selector': 'tr:hover', 'props': [('background-color', 'white'),
('border', '1.5px solid black')]}])
```

	outlook	temp	humidity	windy	play
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes
5	rainy	cool	normal	True	no
9	rainy	mild	normal	False	yes
13	rainy	mild	high	True	no

```
print(f'Entropy of the Rainy dataset: {find_entropy(rainy.play)}')
```

```
    Entropy of the Rainy dataset: 0.971
```

```
#temp
entropy_and_infogain(rainy, 'temp')
```

	temp	play
3	mild	yes
9	mild	yes
13	mild	no

```
Entropy of temp - mild = 0.918
```

	temp	play
4	cool	yes
5	cool	no

```
Entropy of temp - cool = 1.0
Information Gain for temp = 0.02
```

```
#Humidity
entropy_and_infogain(rainy, 'humidity')
```

	humidity	play
3	high	yes
13	high	no

```
Entropy of humidity - high = 1.0
```

	humidity	play
4	normal	yes
5	normal	no
9	normal	yes

```
Entropy of humidity - normal = 0.918
Information Gain for humidity = 0.02
```

```
#Windy
entropy_and_infogain(rainy, 'windy')
```

	windy	play
3	False	yes
4	False	yes
9	False	yes

```
Entropy of windy - False = 0.0
```

	windy	play
5	True	no
13	True	no

```
Entropy of windy - True = 0.0
Information Gain for windy = 0.971
```

#Making a decision tree node using the feature which has the maximum Information Gain.