Program 1

Write a python program to import and export data using pandas library functions?

Code:

Import

```
import pandas as pd

airbnb_data = pd.read_csv("listings (1).csv")

airbnb_data.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_r
0	329172	Hillside designer home,10 min.dwntn	1680871	Janet	NaN	78746	30.30085	-97.80794	Entire home/apt	495	3	7	2022-
1	329306	Urban Homestead, 5 minutes to downtown	880571	Angel	NaN	78702	30.27232	-97.72579	Private room	63	2	570	2022-
2	331549	One Room with Private Bathroom	1690383	Sandra	NaN	78725	30.23911	-97.58625	Private room	100	2	0	
3	333815	Solar Sanctuary - Austin Room	372962	Kim	NaN	78704	30.25381	-97.75262	Private room	102	2	164	2022-
4	333442	Rare Secluded 1940s Estate	1698318	Virginia	NaN	78703	30.31267	-97.76641	Entire home/apt	286	3	163	2022-

Export

```
airbnb_data.to_csv("list2.csv")
```

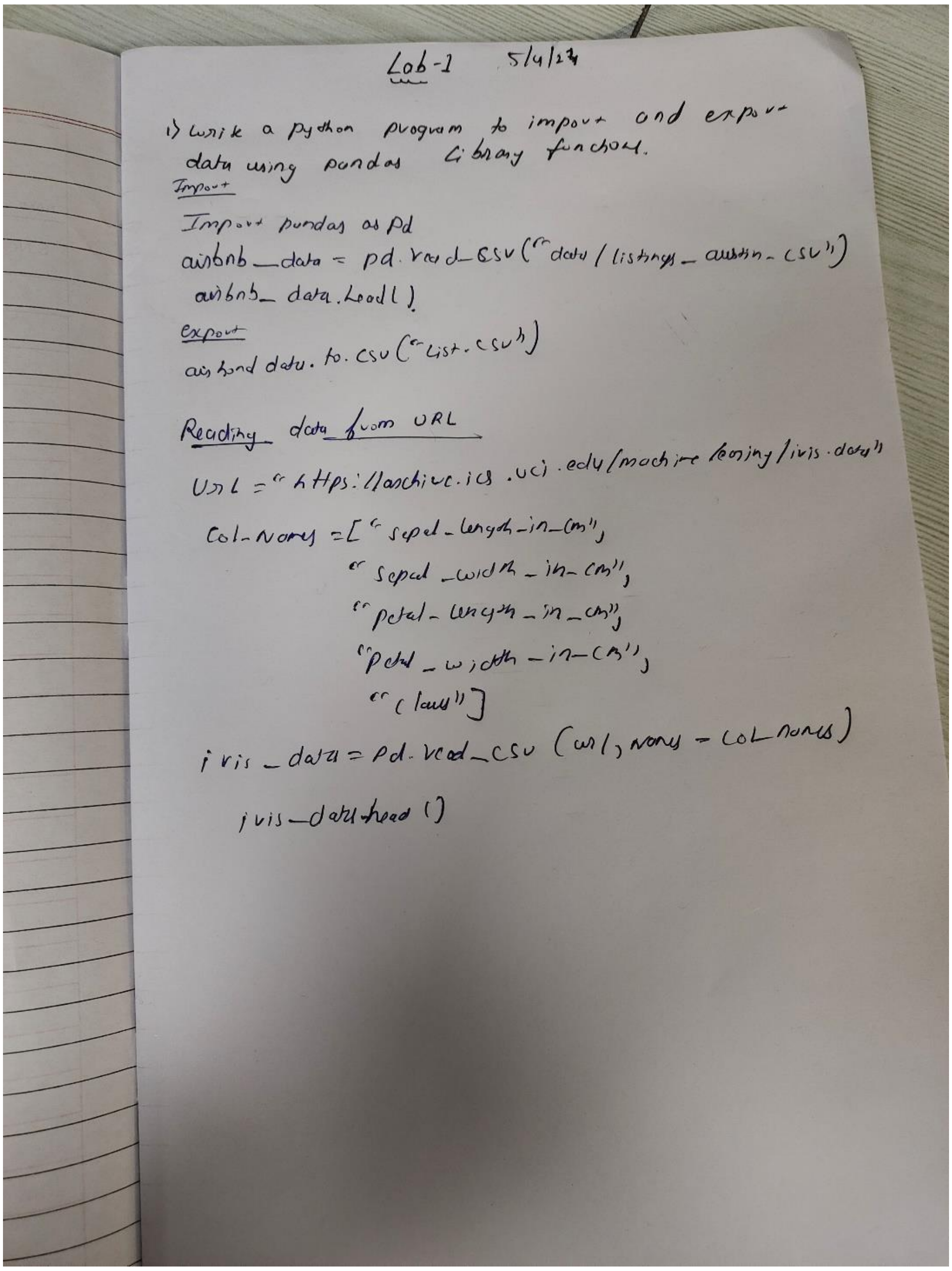
Reading the file from the URL:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

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()
```

2]:

	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

Screenshot from the lab record:



Lab-1 5/4/24

1) write a python program to import and export data using pandas library functions.

Import

```
Import pandas as pd
airbnb_data = pd.read_csv("data/listings_austin.csv")
airbnb_data.head()
```

export

```
airbnb data.to.csv("List.csv")
```

Reading data from URL

```
Url = "https://archive.ics.uci.edu/machine learning/iris.data"
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 ()
```

Program 2

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

1.importing database

```
[3] import pandas as pd
    from sklearn.tree import DecisionTreeClassifier, plot_tree
    import matplotlib.pyplot as plt
    import math
```

```
[4] df = pd.read_csv('/content/diabetes.csv')
    df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

2.Calculating entropy and information gain.

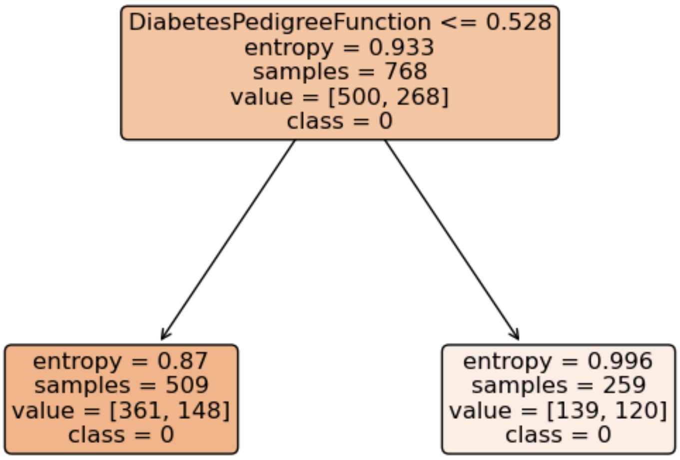
```
def calculate_entropy(data, target_column): # for each categorical variable
  total_rows = len(data)
  target_values = data[target_column].unique()

  entropy = 0
  for value in target_values:
    # Calculate the proportion of instances with the current value
    value_count = len(data[data[target_column] == value])
    proportion = value_count / total_rows
    entropy -= proportion * math.log2(proportion) if proportion != 0 else 0

  return entropy

def calculate_information_gain(data, feature, target_column):

  # Calculate weighted average entropy for the feature
  unique_values = data[feature].unique()
  weighted_entropy = 0

  for value in unique_values:
    subset = data[data[feature] == value]
    proportion = len(subset) / len(data)
    weighted_entropy += proportion * calculate_entropy(subset, target_column)

  # Calculate information gain
  information_gain = entropy_outcome - weighted_entropy

  return information_gain
```

```
[19] for column in df.columns[:-1]:
       entropy = calculate_entropy(df, column)
       information_gain = calculate_information_gain(df, column, 'Outcome')
       print(f"{column} - Entropy: {entropy:.3f}, Information Gain: {information_gain:.3f}")
```

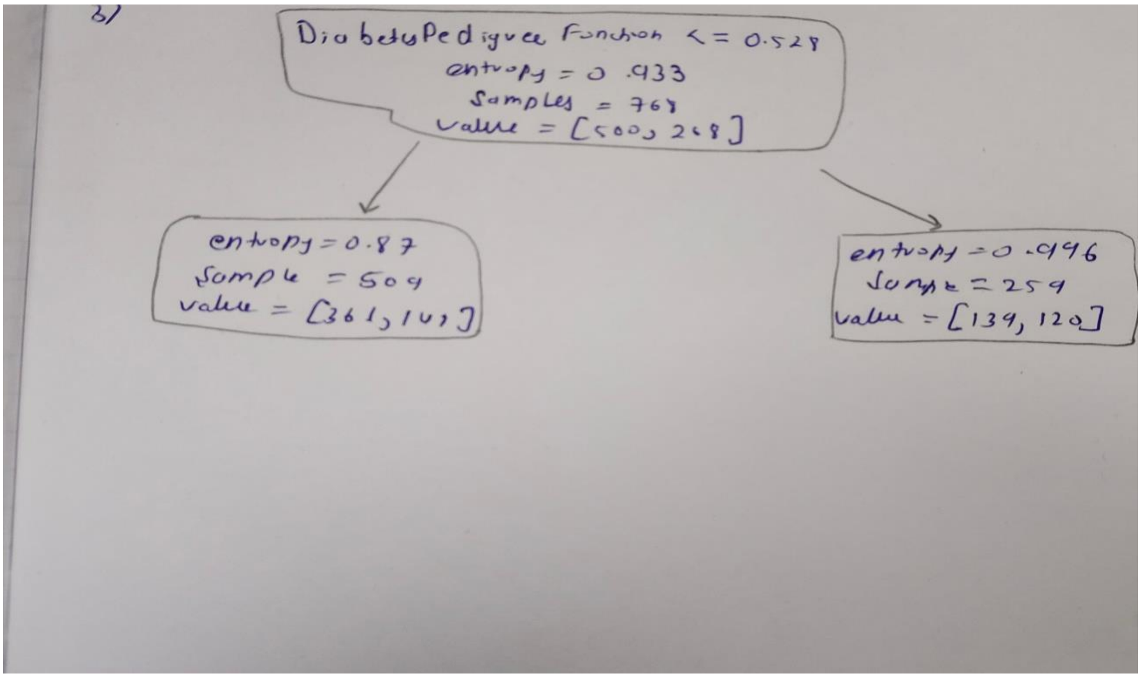
```
Pregnancies - Entropy: 3.482, Information Gain: 0.062
Glucose - Entropy: 6.751, Information Gain: 0.304
BloodPressure - Entropy: 4.792, Information Gain: 0.059
SkinThickness - Entropy: 4.586, Information Gain: 0.082
Insulin - Entropy: 4.682, Information Gain: 0.277
BMI - Entropy: 7.594, Information Gain: 0.344
DiabetesPedigreeFunction - Entropy: 8.829, Information Gain: 0.651
Age - Entropy: 5.029, Information Gain: 0.141
```

3.Making Decision tree.

```python
# Feature selection for the first step in making decision tree
selected_feature = 'DiabetesPedigreeFunction'

# Create a decision tree
clf = DecisionTreeClassifier(criterion='entropy', max_depth=1)
X = df[[selected_feature]]
y = df['Outcome']
clf.fit(X, y)

plt.figure(figsize=(8, 6))
plot_tree(clf, feature_names=[selected_feature], class_names=['0', '1'], filled=True, rounded=True)
plt.show()
```



4.snapshot.



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

ID3 - algorithm

1. Determine entropy for the overall the dataset using class distribution
2. For each feature
   - Calculate Entropy for categorical value.
   - Assess information gain for each unique
3. Choose the feature that generated highest information gain
4. Iteratively apply all above steps to build the decision tree structure.

Output

1) Entropy of the dataset :- 0.43313

2) calculating Entropy & IG for each case.

		Entropy	IG
Pregnancies	-	3.48	0.062
Glucose	-	6.75	0.304
Bloodpressure	-	4.79	0.059
Skin Thickness	-	4.58	0.082
Insulin	-	4.68	0.277
BMI	-	7.59	0.344
Diabetes	-	8.82	0.651 — Highest IG
Age	-	5.02	0.141