Assignment 2: Plotting

Solve the following exercises and upload your solutions to Moodle by the due date.

Important Information!

Please try to *exactly match the outputs* provided in the examples.

Use the *exact filenames* specified for each exercise (the default suggestions from the heading). Your main code (example prints, etc.) should be guarded by `if __name__ == '__main__':`. Unless explicitly stated otherwise, you can assume correct user input and correct arguments.

You may use **only standard libraries**, plus modules covered in the lecture. NumPy was done in Python 1 and can also be used.

Exercise 1 – Submission: `a2_ex1.py` 25 Points

Write a function `plot_runtime(data: dict, figsize: tuple, save_path: str = None)` that plots the runtime of two algorithms (`Algorithm 1`, `Algorithm 2`) with respect to the number of instances of input datasets.

The function parameters are:

- `data` is a dictionary that contains items as follows:

```
{
    'n_instances':n_instances,
    'Algorithm 1': runtime_1,
    'Algorithm 2': runtime_2
}
- n_instances is a list of instance numbers of testing datasets.
- runtime_1 and runtime_2 are, respectively, lists of runtime of Algorithm 1
  and Algorithm 2 w.r.t. testing datasets.
```

- `figsize` is a tuple of two integers (width, height) that indicates the size of the figure of the plotting figure.
- `save_path` is a valid file path to save the plotting figure. The function will not store the figure if `save_path` is None.

The function has to create the following line plot using `matplotlib` as shown in Figure 1:

- plot two lines for the runtime of the two algorithms as in Figure 1:
    - the runtime on y-axis and the instance numbers on x-axis
    - line styles: '-'
    - colors: red and blue
    - makers: square and circle
- set the labels for x and y axes as in Figure 1
- set the title for the plot as in Figure 1
- set the legend for the plot as in Figure 1
- show the grid for both x and y axes as follows:
    - line styles: '--'
    - alpha: 0.6

- keep the auto-generated y-ticks by `matplotlib`.
- apply new x-ticks by converting the number of instances in `n_instances`, e.g. $x000000 \longrightarrow xM$
- apply the tight layout for the figure.

The following code produces a plot as in Figure 1 (example data in `a2_ex1_data.pkl`):

```
with open("a2_ex1_data.pkl", "rb") as f:
    data = pkl.load(f)
plot_runtime(data, (7, 4), "ex1.png")
```

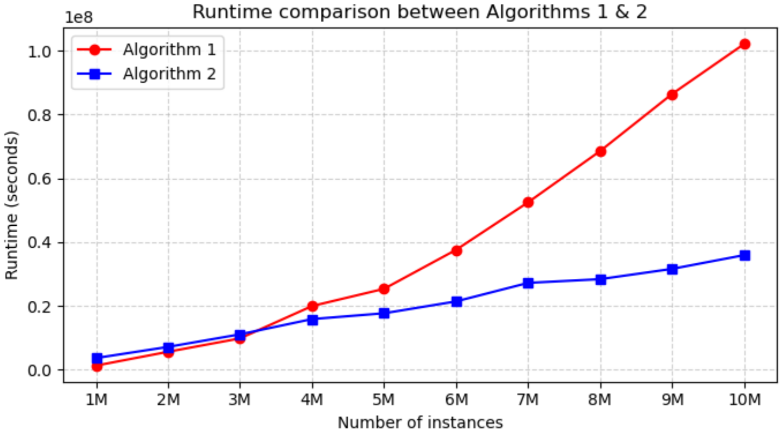

Figure 1: Runtime of Algorithm 1 and Algorithm 2

Exercise 2 – Submission: `a2_ex2.py` 25 Points

Write a function `plot_scores(data: np.ndarray, figsize: tuple, save_path: str = None)` that shows the distribution of exam scores of students from a course.

The function parameters are:

- `data` is an 1D float `numpy` array.
- `figsize` is a tuple of two integers (width, height) that indicates the size of the figure of the plotting figure.
- `save_path` is a valid file path to save the plotting figure. The function will not store the figure if `save_path` is None.

The function creates two histogram plots (left and right sides) for the exam scores using `matplotlib` as shown in Figure 2:

- plot the left-side histogram:
  - uses custom bins [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
  - color: blue
  - edge color: black
- plot the right-side histogram:
  - uses custom bins [0, 50, 62.5, 75, 87.5, 100]
  - color: red
  - edge color: black
  - apply new x-ticks ["Unsatisfactory", "Adequate", "Satisfactory", "Good", "Very Good"], the x-ticks are rotated 45 degree and put at the middle of each bin as in Figure 2.
- set the titles for the two plots as in Figure 2, ’500’ is a dynamic value which is the number of scores.
- y-axis is shared for both plots and uses the auto-generated y-ticks.
- set the labels for x and y axes as in Figure 2.
- show the y-axis grid for both plots as follows:
  - line styles: ’–’
  - alpha: 0.6
- apply the tight layout for the figure.

The following code produces the two histogram plots as in Figure 2 (example data in `a2_ex2_data.csv`):

```
data = np.loadtxt("a2_ex2_data.csv", delimiter=",", skiprows=1)
plot_scores(data, (10, 5), "ex2.png")
```

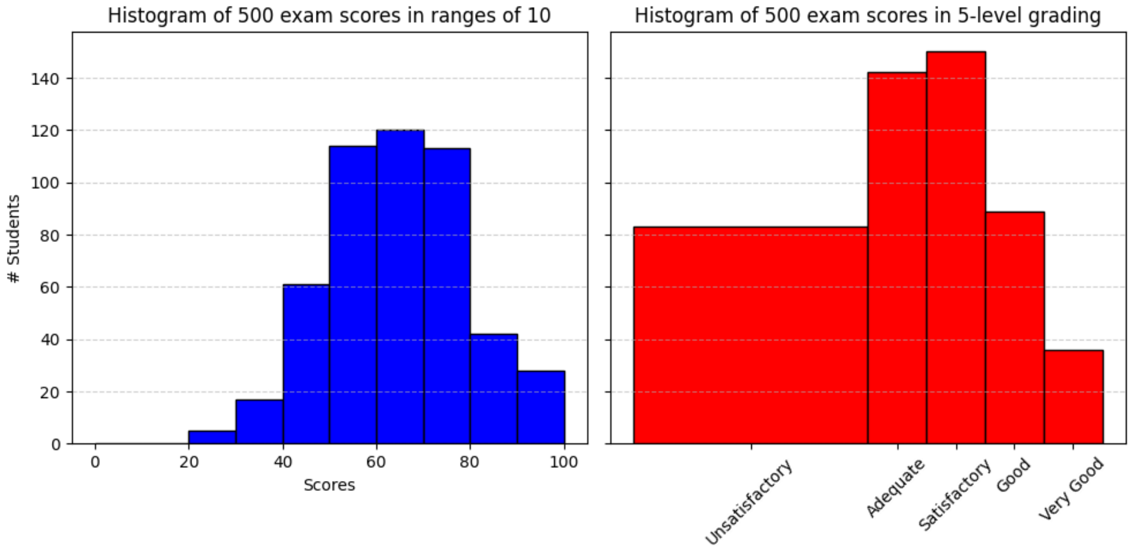

Figure 2: Exam score distribution

Exercise 3 – Submission: `a2_ex3.py` **20 Points**

Write a function `plot_distribution(data: dict, figsize: tuple, save_path: str = None)` that plots the distribution of three kinds of plants (`Plant A, B and C`) in an area.

The function parameters are:

- `data` is a dictionary that contains items as follows:

```
{
'Plant A': plant_a,
'Plant B': plant_b,
'Plant C': plant_c
}
- plant_a, plant_b, plant_c are three 2D float numpy arrays,
  shape(N, 2), which contain the coordinates of instances of plants
  A, B and C respectively. N is the number of plant instances.
```

- `figsize` is a tuple of two integers (width, height) that indicates the size of the plotting figure.
- `save_path` is a valid file path to save the plotting figure. The function will not store the figure if `save_path` is None.

The function creates the following scatter plot using `matplotlib` as shown in Figure 3:

- Plant A:
  - marker: circle
  - colors: red
  - alpha: 0.4
- Plant B:
  - marker: 'x'
  - colors: green
  - alpha: 0.4

- Plant C:
  - marker: square
  - colors: blue
  - alpha: 0.4
- set the labels for x and y axes as in Figure 3
- set the title for the plot as in Figure 3
- set the legend for the plot as in Figure 3. The values 500, 250, and 200 are dynamic values which are the number of instances of plants A, B, and C respectively.
- keep the auto-generated x-ticks and y-ticks by `matplotlib`.
- apply the tight layout for the figure.

The following code produces a plot as in Figure 3 (example data in `a2_ex3_data.pkl`):

```
with open("a2_ex3_data.pkl", "rb") as f:
    data = pkl.load(f)
plot_distribution(data, (10, 6), "ex3.png")
```

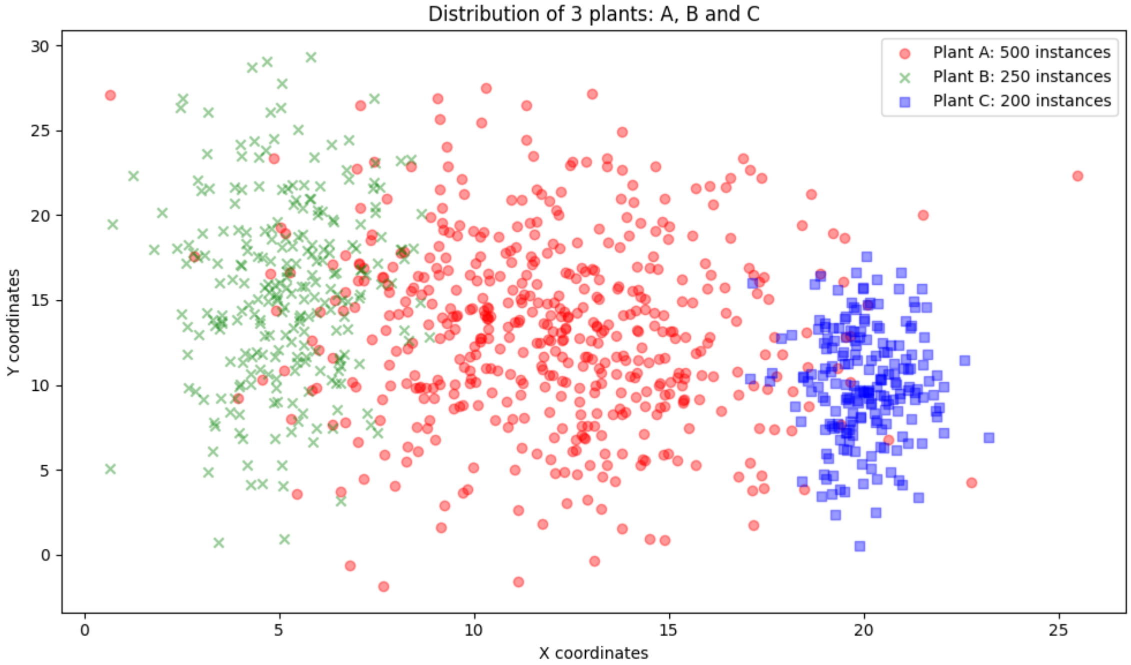

Figure 3: The distribution of plants A, B and C in an area

Exercise 4 – Submission: `a2_ex4.py` 30 Points

Write a function `plot_evaluation_metrics(data: dict, figsize: tuple, save_path: str = None)` that plots two evaluation metrics, Accuracy and F1-score, for n models.

The function parameters are:

- `data` is a dictionary that contains items as follows:

```
{
"Accuracy": {
    "values": acc_values,
    "labels": labels
},
"F1": {
    "values": f1_values,
    "labels": labels
}
- acc_values is a list of n 1D float numpy arrays. Each array contains the
  accuracy values of a model.
- f1_values is a list of n 1D float numpy arrays. Each array contains the
  f1-score values of a model.
- labels is a list of names of n models.
```

- `figsize` is a tuple of two integers (width, height) that indicates the size of the plotting figure.
- `save_path` is a valid file path to save the plotting figure. The function will not store the figure if `save_path` is None.

The function creates the following box plot using `matplotlib` as shown in Figure 4:

- for each evaluation metric in separate columns, a box plot showing data of the n arrays in distinct colors according to the fixed colormap, using color map ’tab10’.
- the box plots must be drawn vertically.
- the median line (enabled per default) must be set to the color black.
- the x-ticks of each evaluation metric box plot must use the model names.
- the y-axis is shared across all plots, and the y-ticks must range from 0.0 to 1.0 (inclusive) with a step size of 0.1. Additionally, make sure that the plots include a space of 0.05 below and above 0.0 and 1.0, respectively, i.e., the borders/limits of the plot should not be 0.0 and 1.0.
- the title of each column/box plot must be set to the corresponding evaluation metric name.
- show the y-axis grid for each plot as follows:
  - line styles: ’–’
  - alpha: 0.6
- apply the tight layout for the figure.

The following code produces a plot as in Figure 4 (example data in `a2_ex4_data.pkl`):

```
with open("a2_ex4_data.pkl", "rb") as f:
    data = pkl.load(f)
plot_evaluation_metrics(data, (3 * len(data), 4), "ex4.png")
```

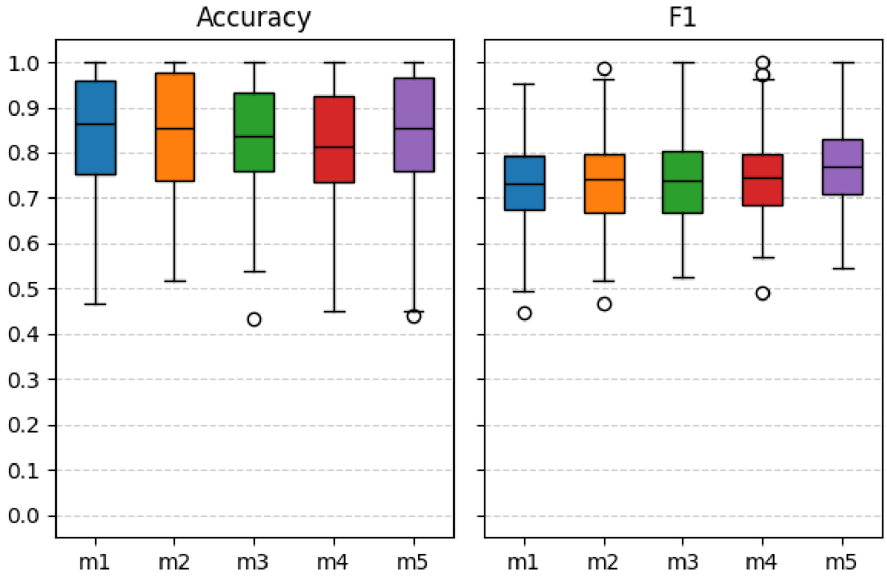

Figure 4: Evaluation metrics of models