Multi-cell-type classification on PBMC-33K dataset

In [1]:
```
%load_ext autoreload
%autoreload 2
%config Completer.use_jedi = False
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import umap

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import *
```

1. Load dataset

In [4]:
```
## Let's load the filtered dataset as anndata object
X = np.load('../data/pbmc_33k/33k_multi_Tcells.npy',  mmap_mode='r')
y = np.load('../data/pbmc_33k/33k_multi_Tcells_lbl.npy', mmap_mode='r')
gene_names = pd.read_csv('../data/pbmc_33k/33k_gene_ids.csv')
```

In [5]:
```
target_names = ['Tcm/Naive cytotoxic T cells', 'Tcm/Naive helper T cells'
                'Tem/Effector helper T cells', 'Tem/Trm cytotoxic T cells
```

In [6]:
```
X.shape
```

Out[6]:
```
(8992, 32738)
```

In [7]:
```
set(y)
```

Out[7]:
```
{0, 1, 2, 3}
```

In [8]:
```
# making sure the length of the data and labels match
assert X.shape[0] == len(y)
```

2. Task

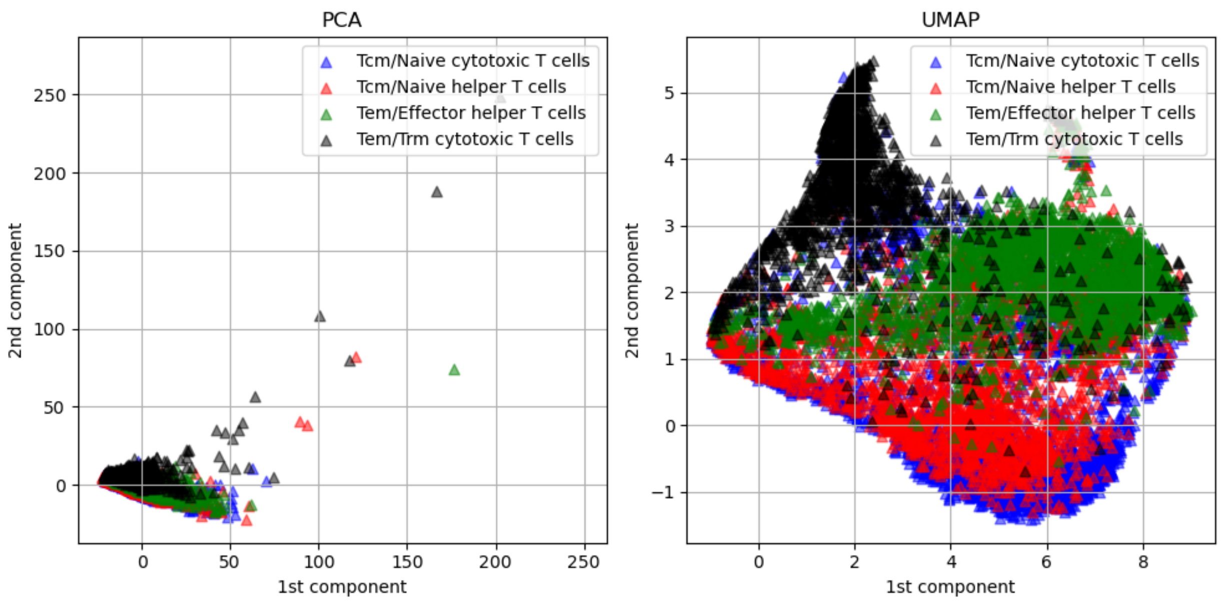
In this tutorial we will again work with PBMC-33K dataset which is now filtered to four cell types: Tcm/Naive cytotoxic T cells, Tem/Trm cytotoxic T cells, Tem/Effector helper T cells and Tcm/Naive helper T cells. List of markers for T-cells

1. Please first visualize how data looks like now compared to binary case using PCA and UMAP. Use data pre-processing steps when needed.

2. Then compare the following algorithms on how well they differentiate between four classes of T-cells. You can use reduced PCA=100 as your input to accelarate training. Please generate following plots to illustrate perfomance comparison.

   - Logistic Regression
   - KNN (k=3)
   - KNN (k=10)
   - Decision Tree

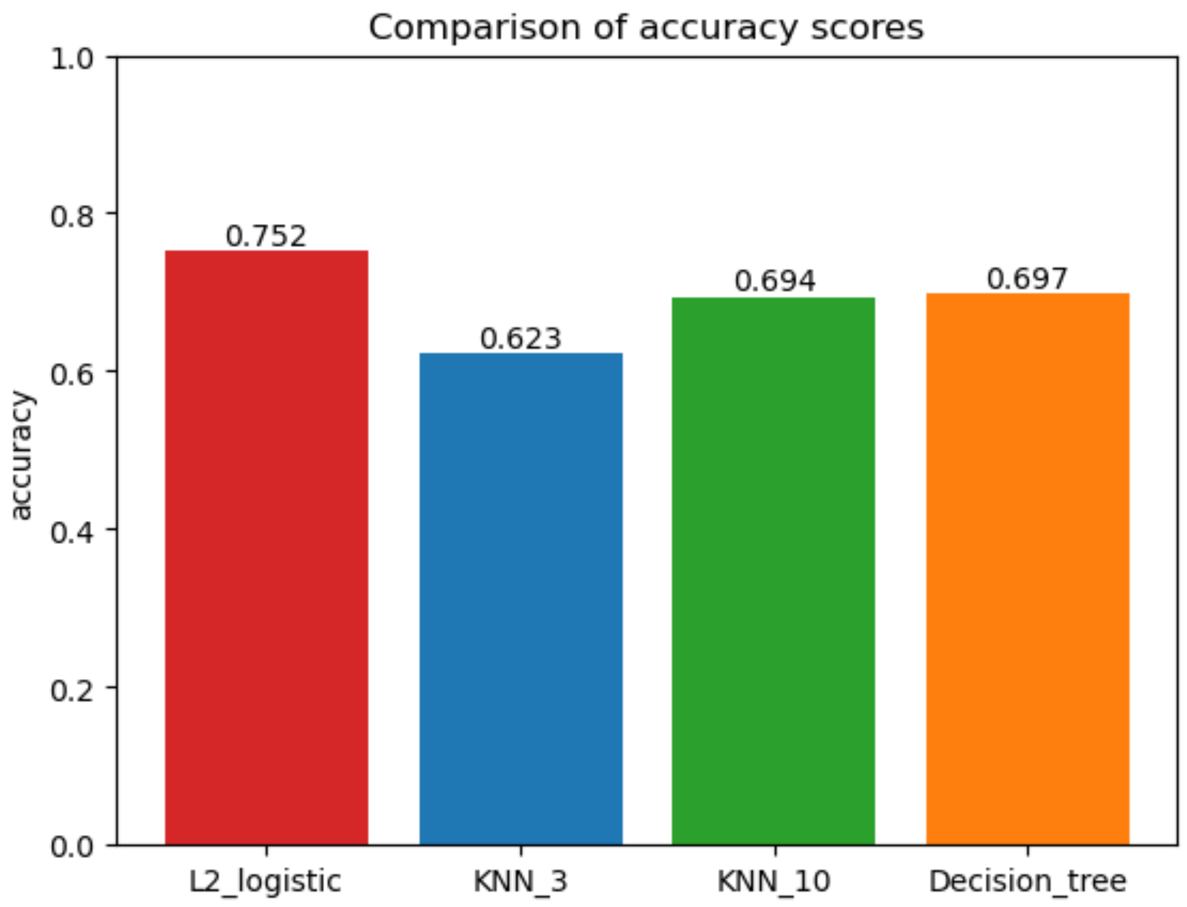
In [15]:
```
### PCA & UMAP
```



In [20]:
```
### Comparison of accuracy scores




#### ps. don't worry if you don't reproduce the exact values
```

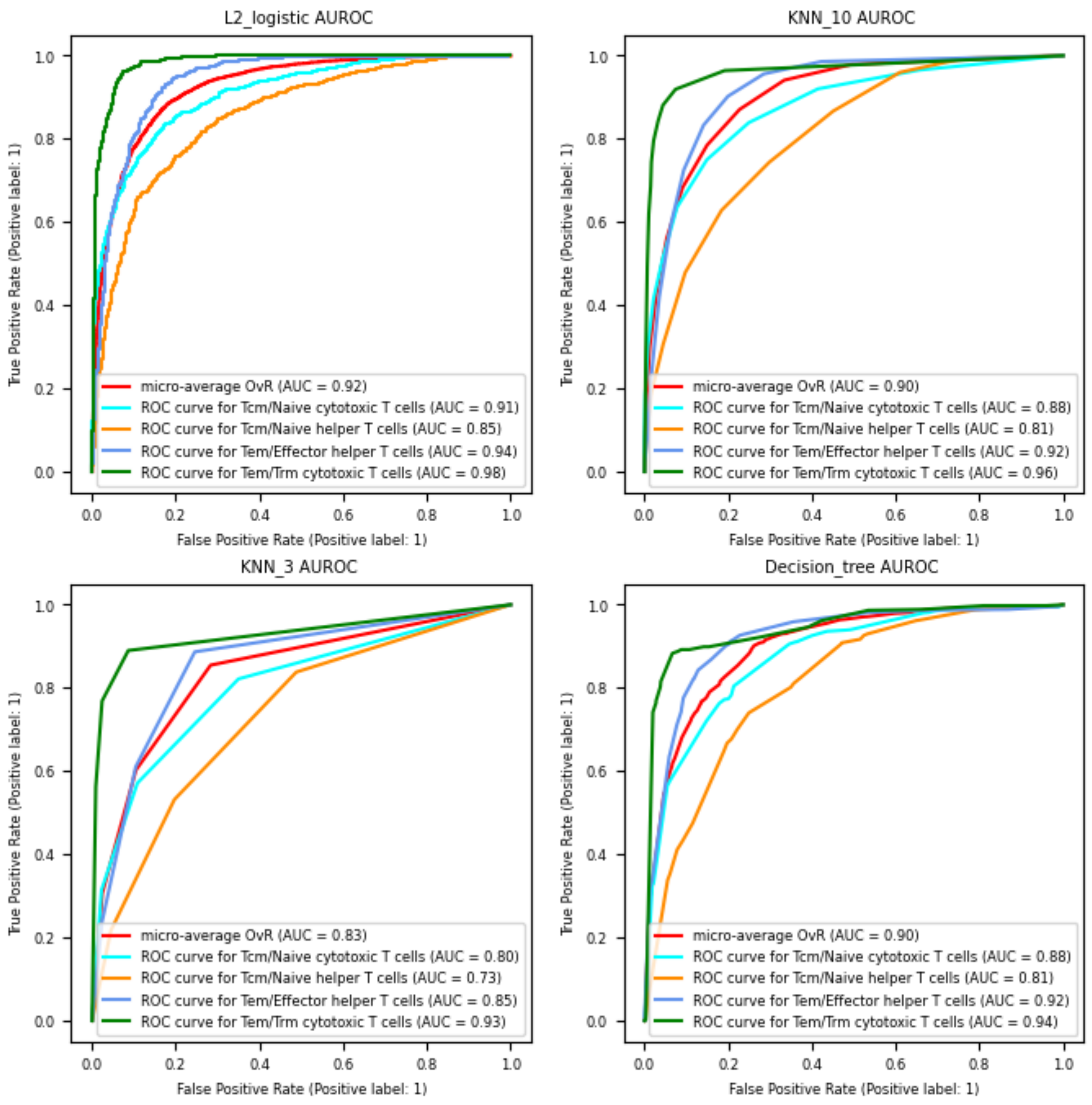


#

Hint: You can make use of `from sklearn.preprocessing import LabelBinarizer` and `RocCurveDisplay.from_predictions` functions to generate the following plots.

In [24]:
```
fig, axs = plt.subplots(2, 2, figsize=(8, 8))
matplotlib.rcParams.update({'font.size': 6})
colors = ("aqua", "darkorange", "cornflowerblue", "green")

#### AUROC plots
```



In [ ]:
```
## feel free to explore how different hyperparameters have an effect on m
# e.g. max_depth for DecisionTree classifier.
```

In [ ]:
```
# the end :)
```