CS 391LH **Spring 2024**

Homework 0 - Programming

Lecture: Prof. Adam Klivans

Keywords: Jupyter, scikit-learn

I recommend you use Google Colab for this course. Go to the Google Colab website and try opening up a new notebook. If you want to run locally that's fine: we will use Jupyter Notebook with Python 3. Install Python 3.8, jupyter, scikit-learn, pandas, and matplotlib. We also recommend installing seaborn, a data visualization library. You may find it convenient to use Anaconda to manage your Python setup.

Load the Wisconsin breast cancer data set, which is available as part of the sklearn datasets. You can read more about this dataset here.

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

The object `cancer` is a dictionary, you can check the keys using

```
cancer.keys()
```

and check the details by calling the keys, like by using

```
cancer.feature_names
```

to see the feature names. You can also view a more detailed description of the dataset:

```
print(cancer.DESCR)
```

Please read the description! On your own, explore the other functions related to the keys. You will also want to load the data (from `cancer.data`) into a pandas DataFrame.

```
df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
```

You can select the column(s) of the DataFrame by indexing on the column name(s). E.g., to see the 'mean radius' column of `df`, simply type

```
df['mean radius']
```

Notice that invoking the above command returns a Series, a fundamental Pandas data-structure. You can read more about this online.

You can select the row(s) of the DataFrame in multiple ways. One way is by slicing. For example, if you want to see the top 100 rows of `df`, simply type

```
df[:100]
```

Or if you want to see the 100th, simply type

```
df[99:100]
```

Another way would be

```
df.iloc[99]
```

You can read more about the `iloc` method online.

The questions below involve exploring the data by plotting some figures, and you will find libraries like `numpy`, `matplotlib` and `seaborn` very useful. You may want to read up on built in functions like `where` from `numpy` for finding indices (or `index` from `pandas`) and `show`, `savefig` from `matplotlib` to show and save your plots.

(a) How many rows and columns are there in this data set? What do the rows and columns represent?

(b) How many malignant cases are there in total?

(c) Make a scatterplot of ‘mean compactness’ and ‘mean perimeter’. Try to color the points by their label (i.e. malignant or benign). You may find `scatter` (from matplotlib) or `jointplot` (from seaborn) useful for this.

(d) Make pairwise scatterplots of ‘mean texture’, ‘mean perimeter’, ‘mean compactness’, and ‘mean radius’. You may find `scatter_matrix` (from matplotlib) or `pairplot` (from seaborn) useful for this. Do you notice that two of these features seem to be particularly correlated? Why might this be?