AI for Medicine Course 1 Week 1 lecture exercises

Patient Overlap and Data Leakage

Patient overlap in medical data is a part of a more general problem in machine learning called **data leakage**. To identify patient overlap in this week's graded assignment, you'll check to see if a patient's ID appears in both the training set and the test set. You should also verify that you don't have patient overlap in the training and validation sets, which is what you'll do here.

Below is a simple example showing how you can check for and remove patient overlap in your training and validations sets.

In [1]:
```
# Import necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import os
import seaborn as sns
sns.set()
```

Read in the data from a csv file

First, you'll read in your training and validation datasets from csv files. Run the next two cells to read these csvs into `pandas` dataframes.

In [2]:
```
# Read csv file containing training data
train_df = pd.read_csv("nih/train-small.csv")
# Print first 5 rows
print(f'There are {train_df.shape[0]} rows and {train_df.shape[1]} co
train_df.head()
```

```
There are 1000 rows and 16 columns in the training dataframe
```

Out[2]:

	Image	Atelectasis	Cardiomegaly	Consolidation	Edema	Effusion	Emphysema
0	00008270_015.png	0	0	0	0	0	0
1	00029855_001.png	1	0	0	0	1	0
2	00001297_000.png	0	0	0	0	0	0
3	00012359_002.png	0	0	0	0	0	0
4	00017951_001.png	0	0	0	0	0	0

In [3]:
```
# Read csv file containing validation data
valid_df = pd.read_csv("nih/valid-small.csv")
# Print first 5 rows
print(f'There are {valid_df.shape[0]} rows and {valid_df.shape[1]} co
valid_df.head()
```

```
There are 109 rows and 16 columns in the validation dataframe
```

Out[3]:

	Image	Atelectasis	Cardiomegaly	Consolidation	Edema	Effusion	Emphysema
0	00027623_007.png	0	0	0	1	1	0
1	00028214_000.png	0	0	0	0	0	0
2	00022764_014.png	0	0	0	0	0	0
3	00020649_001.png	1	0	0	0	1	0
4	00022283_023.png	0	0	0	0	0	0

Extract and compare the PatientId columns from the train and validation sets

By running the next four cells you will do the following:

1. Extract patient IDs from the train and validation sets
2. Convert these arrays of numbers into `set()` datatypes for easy comparison
3. Identify patient overlap in the intersection of the two sets

In [4]:
```
# Extract patient id's for the training set
ids_train = train_df.PatientId.values
# Extract patient id's for the validation set
ids_valid = valid_df.PatientId.values
```

In [5]:
```
# Create a "set" datastructure of the training set id's to identify u
ids_train_set = set(ids_train)
print(f'There are {len(ids_train_set)} unique Patient IDs in the trai
# Create a "set" datastructure of the validation set id's to identify
ids_valid_set = set(ids_valid)
print(f'There are {len(ids_valid_set)} unique Patient IDs in the vali
```

```
There are 928 unique Patient IDs in the training set
There are 97 unique Patient IDs in the validation set
```

In [6]:
```
# Identify patient overlap by looking at the intersection between the
patient_overlap = list(ids_train_set.intersection(ids_valid_set))
n_overlap = len(patient_overlap)
print(f'There are {n_overlap} Patient IDs in both the training and va
print('')
print(f'These patients are in both the training and validation datase
print(f'{patient_overlap}')
```

```
There are 11 Patient IDs in both the training and validation sets

These patients are in both the training and validation datasets:
[20290, 27618, 9925, 10888, 22764, 19981, 18253, 4461, 28208, 876
0, 7482]
```

Identify rows (indices) of overlapping patients and remove from either the train or validation set

Run the next two cells to do the following:

1. Create lists of the overlapping row numbers in both the training and validation sets.
2. Drop the overlapping patient records from the validation set (could also choose to drop from train set)

In [7]:
```
train_overlap_idxs = []
valid_overlap_idxs = []
for idx in range(n_overlap):
    train_overlap_idxs.extend(train_df.index[train_df['PatientId'] ==
    valid_overlap_idxs.extend(valid_df.index[valid_df['PatientId'] ==

print(f'These are the indices of overlapping patients in the training
print(f'{train_overlap_idxs}')
print(f'These are the indices of overlapping patients in the validati
print(f'{valid_overlap_idxs}')
```

```
These are the indices of overlapping patients in the training set
:
[306, 186, 797, 98, 408, 917, 327, 913, 10, 51, 276]
These are the indices of overlapping patients in the validation s
et:
[104, 88, 65, 13, 2, 41, 56, 70, 26, 75, 20, 52, 55]
```

In [8]:
```
# Drop the overlapping rows from the validation set
valid_df.drop(valid_overlap_idxs, inplace=True)
```

Check that everything worked as planned by rerunning the patient ID comparison between train and validation sets.

When you run the next two cells you should see that there are now fewer records in the validation set and that the overlap problem has been removed!

In [9]:
```
# Extract patient id's for the validation set
ids_valid = valid_df.PatientId.values
# Create a "set" datastructure of the validation set id's to identify
ids_valid_set = set(ids_valid)
print(f'There are {len(ids_valid_set)} unique Patient IDs in the vali
```

```
There are 86 unique Patient IDs in the validation set
```

In [10]:
```
# Identify patient overlap by looking at the intersection between the
patient_overlap = list(ids_train_set.intersection(ids_valid_set))
n_overlap = len(patient_overlap)
print(f'There are {n_overlap} Patient IDs in both the training and va
```

```
There are 0 Patient IDs in both the training and validation sets
```

Congratulations! You removed overlapping patients from the validation set!

You could have just as well removed them from the training set.

Always be sure to check for patient overlap in your train, validation and test sets.

In [ ]: