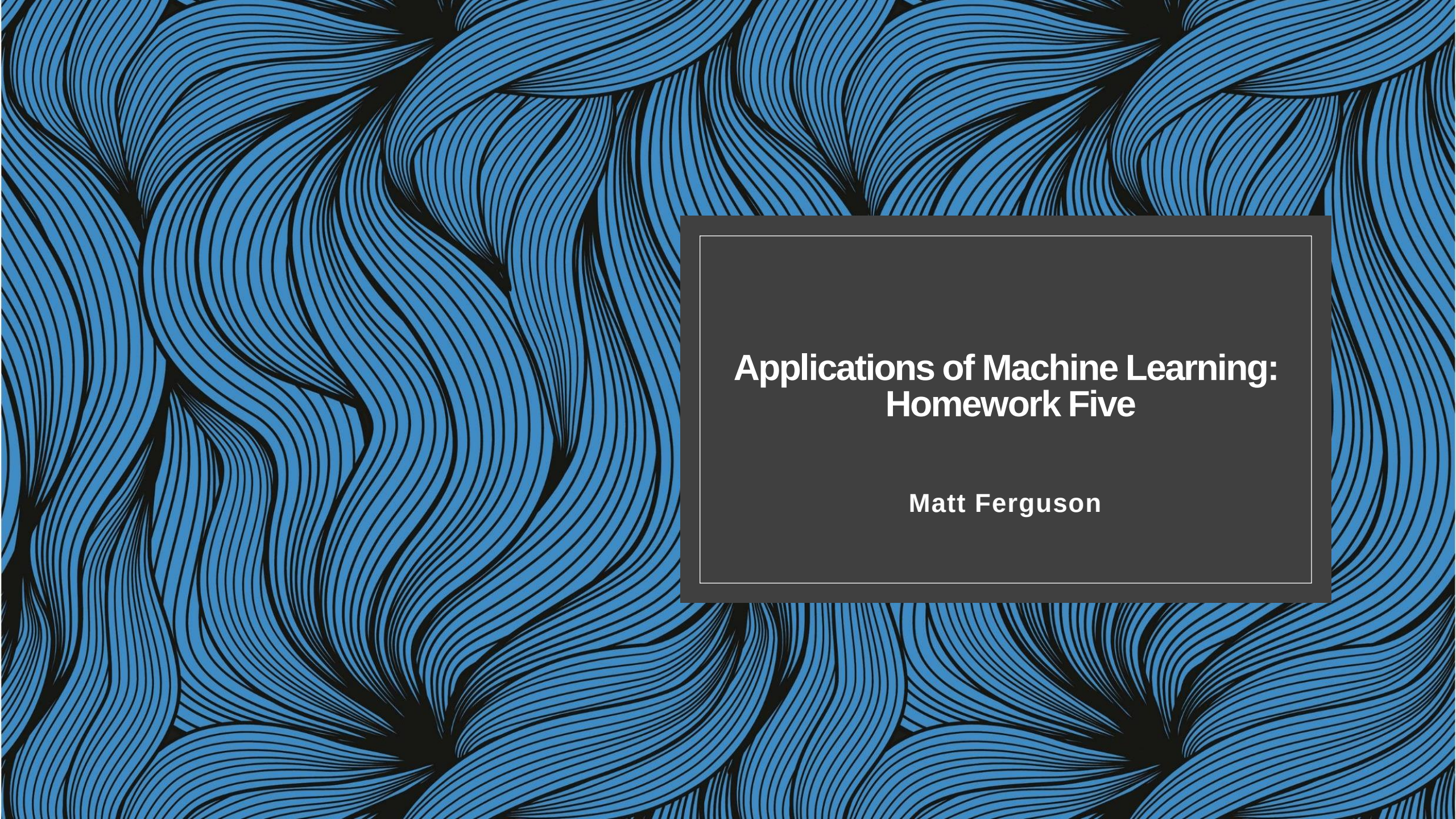

Applications of Machine Learning: Homework Five

Matt Ferguson

Problem Statement

Given a set of combined cycle power plant data investigate the effect of different neural network architectures on classification performance. State which activation functions, and number and sizes of hidden layers, give the best performance.

Requirements

I. Load the provided feature data and binary targets into python, normalize the data and divide the data into training and test sets using a random seed.
II. Train a set of MLP classifier models on the training data. Use early stopping. Iterate through 1-3 hidden layers,1-10 nodes per layer, and all 4 choices for internal activation functions.
III. Measure performance with AUROC and classification accuracy.
    I. Create tables of the best 10 and worst 10 models by AUROC and classification accuracy.
IV. Summarize your results and your findings. Identify the best models and discuss results and observations.
V. Provide the MLP classifier python file.

Features

Temperature (T) in the range 1.81°C and 37.11°C,
Ambient Pressure (AP) in the range 992.89-1033.30 milibar,
Relative Humidity (RH) in the range 25.56% to 100.16%
Exhaust Vacuum (V) in the range 25.36-81.56 cm Hg
Net hourly electrical energy output (EP) 420.26-495.76 MW

EP is our target variable, values for EP above a critical threshold (>450 MW) are marked as 1 and values below are marked as zero. This denotes the outcome that we seek to predict.

Data and feature information is sourced from: https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant

Code

Our code on the left loads the combined cycle power plant dataset into pandas. We read in the sheet ‘allBin’ as this has our target converted to binary.

The data is normalized from 0 to 1 by column using pandas dataframe min and max methods. Then the data is divided into training and test sets using the sklearn model selection method.

We iterate through every combination of layers and nodes. These iterations create a vector to pass to our MLP classifier model that gives the structure of the hidden layer(s) of the neural network. Basically we iterate through every combination of nodes (1-10) and layers (1-3) resulting in a hidden layer vector ranging from (1) – (10,10,10) which creates 1110 possible hidden layer vectors. We also iterate through each of the four activation functions with every hidden layer vector, resulting in a total of 4440 models.

These iterations of hidden layer/activation function are passed as arguments to the MLP classifier for training. Additionally, we train our MLP classifier using early stopping, adaptive learning rate, a tolerance of 0.0001, an adam solver, an alpha of 0.0001, and a validation fraction of 0.1.

Finally after training AUROC and classification accuracy are computed and stored in lists, along with hidden layers, and activation function so we can identify model performance.

We have three nested for loops to iterate through each value of our nodes. Node 2 and Node 3 are allowed to iterate from 0-10 so we have a way of representing layer size 1 or 2. We drop zeroes from our hidden layer vector.

Once we have our hidden layer vector created we concatenate it with the activation function to create a unique key pair and store this to prevent computing duplicate models. Without this step we would have many duplicate models. Imagine the vector (10,0,10), which would be the same as (10,10,0) after dropping of zeroes. If we only use the hidden layer vector for duplicate detection then we would fail to iterate through all activation functions. Therefore the unit for duplication detection is the key pair of hidden layer vector and activation function.

```
import pandas as pd
import numpy as np
import sklearn as sk
from sklearn import neural_network, metrics

filename = (r'C:\Users\Matt\Desktop\ccpp.xlsx')
odf= pd.read_excel(filename)

auroc=[]
accuracy=[]
layers=[]
nodes=[]
functions=[]
sf_check=[]
activation_functions=['identity','logistic', 'tanh', 'relu']

ar=np.array((odf-odf.min())/(odf.max()-odf.min()))
x=ar[:,1:5]
y=ar[:,-1]

(x_train, x_test, y_train, y_test) = sk.model_selection.train_test_split(x, y, test_size=0.3, random_state=22222)

for node1 in range(1,11): #Iterate nodes 1 to 10
    for node2 in range(11):
        for node3 in range(11):
            for function in activation_functions:

                size=[node1,node2,node3]
                size=[node for node in size if node!=0]
                size_and_function=str(size)+function

                if size_and_function not in sf_check:

                    clf = neural_network.MLPClassifier(hidden_layer_sizes=(size), activation=function, learning_rate=
                    'adaptive', tol=0.0001, solver='adam', alpha=0.0001, early_stopping=True, validation_fraction=0.1)

                    clf.fit(x_train, y_train)
                    y_pred=clf.predict(x_test)

                    auroc.append(metrics.roc_auc_score(y_test, y_pred, multi_class='ovr'))
                    accuracy.append(metrics.accuracy_score(y_test, y_pred))
                    layers.append(len(size))
                    nodes.append(size)
                    functions.append(function)

                    sf_check.append(str(size)+function)
```

Best 10 Results and Discussion

On the right we see the 10 best models by classification accuracy and AUROC.

The best performing models have above average node counts. All of the best performing models have a layer of 3, indicating that layer depths of 1 or 2 is strictly inferior in performance. We see 10 of these 20 models (repeats considered) have an ‘identity’ activation function, 8 have the ‘tanh’ function and 2 have the ‘relu’ function. Based on this, we can conclude that ‘identity’ or ‘tanh’ are best for this particular classification problem.

Our average node count for a single layer is 6.5 and our median node count is 7. The mode is 10. Clearly our models with high node and layer counts have high performance metrics. It is quite intuitive that the mode is 10 for node count. More neural network computations would intuitively yield higher accuracy, with diminishing returns. Perhaps higher node counts could be explored to see if accuracy decreases at high node counts and there exists some ‘spooky maximum’.

Hidden Layers	Activation Function	AUROC	**Accuracy**
4, 10, 2	relu	0.9558	0.9551
2, 10, 10	identity	0.9529	0.9523
10, 5, 5	identity	0.9522	0.9516
3, 8, 7	tanh	0.9527	0.9516
7, 9, 5	identity	0.9522	0.9516
1, 6, 5	tanh	0.9526	0.9512
10, 7, 9	tanh	0.9514	0.9512
10, 9, 7	identity	0.9517	0.9512
5, 9, 3	tanh	0.9514	0.9512
9, 4, 7	identity	0.9514	0.9512

Hidden Layers	Activation Function	**AUROC**	Accuracy
4, 10, 2	relu	0.9558	0.9551
2, 10, 10	identity	0.9529	0.9523
3, 8, 7	tanh	0.9527	0.9516
1, 6, 5	tanh	0.9526	0.9512
10, 5, 5	identity	0.9522	0.9516
7, 9, 5	identity	0.9522	0.9516
10, 9, 7	identity	0.9517	0.9512
1, 3, 3	tanh	0.9517	0.9505
7, 8, 9	tanh	0.9515	0.9509
4, 9, 7	identity	0.9515	0.9505

Worst 10 Results and Discussion

On the right we see the worst 10 models by classification accuracy and by AUROC.

The worst performing models seemed to have very low node counts. We see 7 of these 20 models (repeats considered) have a layer length lower than 3. However all 19/20 have a layer with a single node. It seems that in our population of models node sizes of one are extremely detrimental to performance. This is intuitive as a layer with a single node would be a strict bottleneck for our neural network.

In terms of activation functions we have 9 relu, 6 tanh, 3 identity, and 2 logistic. Based on the tables it appears that the rectified linear unit function is a poor choice for our models.

Note that the classification accuracy for our bottom ten models are all equal at 45.7%. In fact 194 models exist all with the same accuracy. The average node size of these models is 4.15 and the median node size is 4. The node mode is 1.

Hidden Layers	Activation Function	AUROC	**Accuracy**
1	logistic	0.5000	0.4566
1	relu	0.5000	0.4566
1, 1	relu	0.5000	0.4566
1, 1, 3	identity	0.5000	0.4566
1, 1, 3	tanh	0.5000	0.4566
1, 1, 4	tanh	0.5000	0.4566
1, 1, 6	relu	0.5000	0.4566
1, 10, 1	logistic	0.5000	0.4566
1, 10, 1	tanh	0.5000	0.4566
1, 10, 2	logistic	0.5000	0.4566

Hidden Layers	Activation Function	**AUROC**	Accuracy
1, 6, 7	relu	0.4937	0.5364
1, 2	tanh	0.4994	0.5427
5, 2, 5	relu	0.4997	0.5430
1	identity	0.5000	0.5434
1	tanh	0.5000	0.5434
2	logistic	0.5000	0.5434
3	identity	0.5000	0.5434
1, 1	logistic	0.5000	0.5434
1, 1	tanh	0.5000	0.5434
1, 1, 1	logistic	0.5000	0.5434