Python Exercises for Petroleum Engineers: Set 2

Welcome to the second homework for your "Introduction to Machine Learning" course. In this assignment, you will be working with a real well-logging dataset. Your task is to build a machine learning model that can predict water saturation using the appropriate set of logs.

In this homework, you will have the opportunity to solve a realistic machine learning problem step by step. To begin, you will need to import your well-logging data into Python and use appropriate tools to describe the data. You will then perform several preprocessing steps, such as

> 1. data cleaning,
> 2. normalization,
> 3. exploratory data analysis,
> 4. and feature selection,

to ensure that the data is ready to be used in a machine learning model.

Next, you should design an ANN model using Keras or Pytorch Lightning libraries. You will then train the neural network with different configurations, including varying the number of layers, number of neurons, learning rates, activation functions, and number of epochs.

During training, you will analyze the loss versus iteration curves to determine the best configuration for your model. You will also need to identify cases of overfitting and underfitting, which can negatively impact the performance of your model. Through this process, you will gain valuable experience in designing and training neural networks, and learn how to optimize their performance for a specific problem.

This homework will provide you with a hands-on experience in solving a realistic machine learning problem and help you develop important skills in data preprocessing, neural network training, and performance analysis. By completing this assignment, you will be well-prepared to tackle more complex machine learning problems in your future career as a petroleum engineer.

So, let's get started and dive into the exercises!

Task 1: Data Import

You have been provided with a well logging data set *FullSet_HW2.las*. This file is an *ASCII* file, which means you can load it by notepad to see what does it contain. Contact me if you had problem with understanding the data file.

Find a way to import this file into Python.

```
In [ ]: # Write your code here:
```

Task 2: Data Description

Describe the provided dataset. Your description should contain

1. shape of the dataset,
2. number of nulls of each column,
3. descriptive statistics (similar to Figure 3-3 "Machine Learning in the Oil and Gas Industry"),
4. and your suggested depth interval to develope the model.

```
In [ ]: # Write your code here:
```

Task 2: Data Preparation

You should perform multiple steps of data preparation through this task. The required data preprocessings are:

1. Data cleaning: In this step you will need to identify and deal with any issues or anomalies in the dataset, to ensure that the data is accurate and consistent. Also you should keep an appropriate interval (a continous interval) of logs and delete the rest of them.

2. Feature selection: You will analyze the data and choose the best set of logs to use as input features for your machine learning model. This step will help you eliminate irrelevant or redundant features, which can negatively impact the performance of your model.

   You should use **your domain knowledge** and **feautre selection methods** to choose the best set of the input logs. Also you should plot correlation heatmap of features and pair plots using seaborn, pandas and/or matplotlib libraries (like Figure 3-3 "Machine Learning in the Oil and Gas Industry")

3. Data normalization

4. Data splitting: Split your data randomly into *Train*, *Test*, and *Validation* sets. Choose a reasonable ratio to split the data.

```
In [ ]: # Write your code here:
```

Task 3: Preparing your ANN code

Prepare a Multi-layer Perceptron (MLP) model with keras or pytorch.

Task 3.1: Prepare your initial architecture

Use the following function (*generate_nn_architecture*) to generate an architecture for you. Implement the generated architecture as a module of the library of your choice (keras or pytorch).

```
import random

def generate_nn_architecture(student_number,initial_input_dim):
    # Set random seed based on student number
    random.seed(student_number)

    # Initialize list to store layers
    nn_architecture = []

    # Generate random number of layers between 3 and 7
    num_layers = random.randint(3, 7)

    # Generate layers
    input_dim = initial_input_dim  # Initial input dimension (number of
features which have you selected in the preprocessing step)
    for i in range(num_layers):
        # Generate random number of neurons between 3 and 5
        output_dim = random.randint(5, 15)
        # Generate random activation function
        activation = random.choice(["relu", "tanh", "sigmoid" , "leaky
relu"])
        # Append layer to architecture list
        nn_architecture.append({"input_dim": input_dim, "output_dim":
output_dim, "activation": activation})
        # Update input dimension for next layer
        input_dim = output_dim

    # Add final layer with sigmoid activation for binary classification
    nn_architecture.append({"input_dim": input_dim, "output_dim": 1,
"activation": "sigmoid"})

    return nn_architecture
```

> *initial_input_dim* value is the number of features that you have selected in the **feature selection** step.

Usage example:

```
student_number = 123456
initial_input_dim = 3 # Number of selected features
nn_architecture = generate_nn_architecture(student_number,
initial_input_dim)
print(nn_architecture)
```

Task 4: Training and analyzing your ANN

Task 4.1: Training your ANN

Consider values for number of epochs and learning rate and train your network.

1. Plot overlay scatter of train cost and validation cost versus epochs,

2. scatter of SWE_pred versus SWE,
3. overlay scatter of SWE_pred versus depth and SWE versus depth.

Task 4.2: Training with different architectures

1. Experiment with different learning rates, activatino functions, number of neurons and number of layers (at least check 20 different setups).
2. Provide the analysis of the performance of different setups you have checked using different metrics and plots (like, parity plot, loss vs. epochs and SWE vs. Depth).
3. Analyze your best and worst found setups and explain your understanding of why they performed such.
4. Show and analyze at least one case with overfitting and one case with underfitting.

****Make sure to follow a systematic path for experimenting differnt setups to facilitate your analysis.****

After completing all the tasks, please, `convert notebooks to PDF`, zip them with your answers notebooks (`ipynb file`) and send it. You should provide sufficient explanation and reporting in your notebooks