National Taiwan University
Department of Civil Engineering

Data Science for Engineering Applications
Instructors: C.-S. CHEN

Python Programming Exam

Due 11:59 am, Wednesday, January 9, 2019

(Take home examination. You **MUST** do it on your own. Discussion with others are strictly prohibited. 一定要自己做，絕對禁止互相討論)

Download the `finalDist.zip` package from the ceiba course website. Once unzipping the file, you will find a folder `finalDist` with two data files to be used. Change the folder name to your ID `b05XXXXXX` (Your ID) and write your solutions in the folder with the name indicated in each problem. You should submit your Jupyter notebook (`*.ipynb`) and Python script (`*.py`). Zip the folder and submit directly on line before the deadline.

1. **(20%)** Name your Jupyter notebook `bostonR2Score` and Python script `bostonR2Score.py`. The Boston housing data set `HousePrice.csv` given in the `finalDist` folder consists of 506 data points with 13 predictors to predict the median house price (`medv`).

(a) (5%) Load the dataset into a panda `DataFrame` object and output the first five records. Below is a sample run:

	crime	zone	indus	river	nox	room	age	dist	highway	tax	ptratio	black	lstat	medv
0	10.0623	0.0	18.1	0	0.584	6.833	94.3	2.0882	24	666	20.2	81.33	19.69	14.1
1	10.6718	0.0	18.1	0	0.740	6.459	94.8	1.9879	24	666	20.2	43.06	23.98	11.8
2	11.1604	0.0	18.1	0	0.740	6.629	94.6	2.1247	24	666	20.2	109.85	23.27	13.4
3	12.0482	0.0	18.1	0	0.614	5.648	87.6	1.9512	24	666	20.2	291.55	14.10	20.8
4	12.2472	0.0	18.1	0	0.584	5.837	59.7	1.9976	24	666	20.2	24.65	15.69	10.2

where the first 13 columns are predictors (`X`) and the last column is the response (`y`) we would like to predict.

(b) (15%) Use all the predictors and split the data points with 70% for training and 30% for testing (`random_state=42`). Fit a second-degree polynomial regression model with the training data and report the R^2 statistics of the training set and testing set. Below is a sample output:

```
Training set R2 statistics: 0.92
Testing set R2 statistics: 0.83
```

2. **(20%)** Name your Jupyter notebook `SimpleRegressionSolution` and Python script `SimpleRegressionSolution.py`. We can implement our own simple linear regression model by knowing:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

where $\hat{y}$ indicates a prediction of Y on the basis of $X = x$. From the least square fitting, we can show:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Implement your own simple linear regression model and compare your results with `Scikit-Learn`. Use the following data to test your implementation.

X	Y
1	9
2	13
3	14
4	18

Below are the sample outputs:

```
From home-made linear regression model
beta0 6.5
beta1 2.8

From Scikit-Learn linear regression model
6.5
[ 2.8]
```

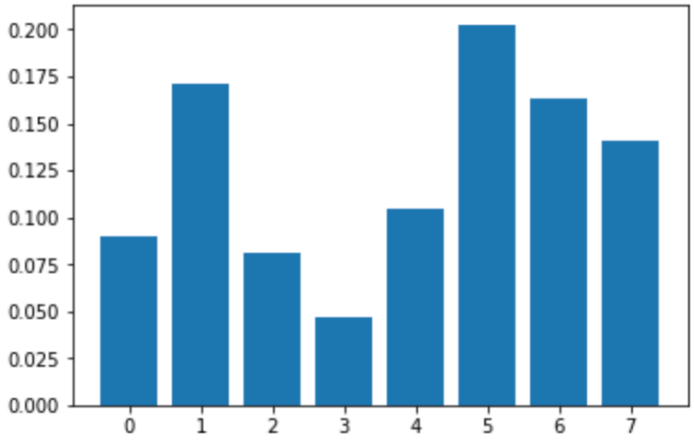
3. **(30%)** Name your Jupyter notebook `xgPima` and Python script `xgPima.py`. The Pima Indians onset of diabetes dataset `pima-indians-diabetes.csv` given in the `finalDist` folder is comprised of 8 input variables that describe medical details of patients and one output variable to indicate whether the patient will have an onset of diabetes within 5 years.
(a) (10%) Run the classification using `XGBClassifier` from `xgboost`. Split 80% for training and 20% for testing with `random_state=42`. Report your accuracy score. Below is a sample output:

```
Accuracy: 75.97%
```

(b) (10%) Refit the model with all the data. Report the feature importance and plot them in a bar chart using `pyplot` from `matplotlib`. Below is a sample output:

```
[0.089701   0.17109634 0.08139535 0.04651163 0.10465116 0.2026578
 0.1627907  0.14119601]
```

```
Out[2]: <BarContainer object of 8 artists>
```



(c) (10%) Use 5-fold cross validation for all the data and report the mean accuracy and standard derivation. Below is a sample output:

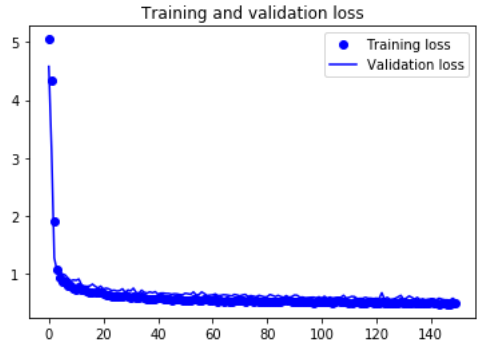
```
Accuracy: mean = 76.57% (std = 3.74%)
```

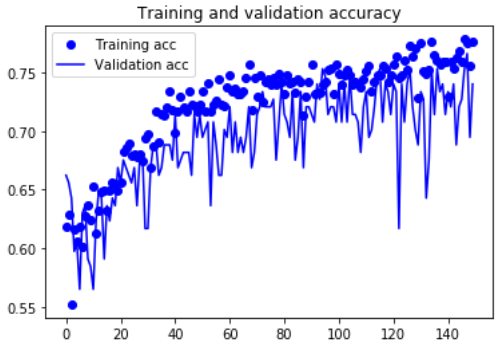
4. (30%) The Pima Indians onset of diabetes dataset `pima-indians-diabetes.csv` given in the `finalDist` folder is comprised of 8 input variables that describe medical details of patients and one output variable to indicate whether the patient will have an onset of diabetes within 5 years.

(a) (10%) Name your Jupyter notebook `nnPima` and Python script `nnPima.py`. Run the classification using a fully-connected neural network structure with three layers. Use the rectifier (`relu`) activation function on the first two layers and the `sigmoid` activation function in the output layer. The first hidden layer has 12 neurons and expects 8 input variables (e.g. input `dim=8`). The second hidden layer has 8 neurons and finally the output layer has 1 neuron to predict the class (onset of diabetes or not). Below is the model summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 12)                108
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 9
=================================================================
Total params: 221
Trainable params: 221
Non-trainable params: 0
```

Specify the `binary_crossentropy` loss function and `adam` optimizer. Split 80% for training and 20% for testing with `random_state=42` using `train_test_split` from `scikit-learn` and report your accuracy score. Fit and validate your model with `150` epochs and a `batch_size` of 10. Plot accuracy and loss figures. Below are the sample plots (your plots might differ from these as randomness is involved in Kera):





(b) (20%) Name your Jupyter notebook `nnCrossPima` and Python script `nnCrossPima.py`. We can obtain a less biased accuracy measurement using cross validation. Use the `StratifiedKFold` class from the `scikit-learn` to split up the training dataset into 5 folds.

```
# define 5-fold cross validation test
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

Again fit and validate your model with `150` epochs and a `batch_size` of 10. Report the evaluation accuracy of each fold and the average and standard deviation of the model performance. Below is a sample output (your numbers might differ from these as randomness is involved in Kera):

```
acc: 72.73%
acc: 68.18%
acc: 68.18%
acc: 65.36%
acc: 67.32%
68.35% (+/- 2.42%)
```