EE258
HOMEWORK #3
(ONLINE SUBMISSION ONLY – MAKE A PDF FILE INCLUDING ALL THE REQUIRED SUBMISSIONS)

DUE: SEP 26th

1. **Read and practice** the codes in Chapter 2 (pages 33-68) of textbook "Hands-On Machine Learning with Scikit-learn and tensorflow" (NOT TO BE SUBMITTED)

2. **Get the data:** For this homework, use the iris dataset under Files-> Datasets on Canvas. (submit the code)

   Hint: Pandas knows to treat rows with 'NA' as missing values

3. **Take a look at the data structure:** Write a brief paragraph about the data set related to each of the following observations (submit the code, the result, and the paragraphs):
   a. Look at the top five rows of the data set
   b. Get a quick description of the data: Notice if there are any missing values or categorical features
   c. Get a summary of the numerical features
   d. Plot the histogram of the numerical features

4. **Discover and visualize the data to gain insights:** (submit the plots, code, and your observations of each plot)
   a. Obtain scatter matrix
   b. Obtain the correlations among features and comment

5. **Data Cleaning** (submit the code, results and plots showing the changes in data)
   a. Drop the data points with NA in it
   b. Tidy up the data by renaming the "class" data point correctly.
   c. Remove the outliers: drop the 'Iris-setosa' rows with a sepal width less than 2.5 cm.

d. One of the data collectors forgot to convert the sepal length for “Iris-versicolor” to cm, instead added the data as meters. Find those and convert them to cm.
e. Handle the categorical variables
f. Save the clean data into a new file.

6. **Utilizing a perceptron learning algorithm to check if a flower is “iris-setosa” or not ?** (submit plots, results, code explainations)
   a. Modify the clean dataset such that you have class =+1: for “iris-setosa”, and class = -1 for others.
   b. Is the new data set linearly separable? Will perceptron algorithm work?
   c. Explain what is the functionality of each line of code in the perceptron.py file.
   d. Separate the data into test and training data
   e. Use the below **perceptron.py** to train your perceptron
   f. Does the algorithm converge? What is niter in the code?
   g. Obtain a plot of the training accuracy as a function of epocs (number of times you go over the entire training data)
   h. Obtain the test data accuracy
   i. Sketch the decision boundary (the line that separates the data; obtained via perceptron algorithm) and the scatter plot of data points. For “iris-setosa” use a different identifier than the rest of the classes.

As performance metric use accuracy:

$$\text{accuracy} = \frac{\text{Number of data points predicted correctly}}{\text{Total number of data points}}$$

```
# perceptron.py
import numpy as np

class Perceptron(object):
   def __init__(self, rate = 0.01, niter = 10):
      self.rate = rate
      self.niter = niter

   def fit(self, X, y):
      """Fit training data
      X : Training vectors, X.shape : [#samples, #features]
      y : Target values, y.shape : [#samples]
      """

      # weights
      self.weight = np.zeros(1 + X.shape[1])

      # Number of misclassifications
      self.errors = []  # Number of misclassifications

      for i in range(self.niter):
         err = 0
         for xi, target in zip(X, y):
            delta_w = self.rate * (target - self.predict(xi))
            self.weight[1:] += delta_w * xi
            self.weight[0] += delta_w
            err += int(delta_w != 0.0)
         self.errors.append(err)
      return self

   def net_input(self, X):
      """Calculate net input"""
      return np.dot(X, self.weight[1:]) + self.weight[0]

   def predict(self, X):
      """Return class label after unit step"""
      return np.where(self.net_input(X) >= 0.0, 1, -1)
```

```
>>> # import Perceptron from perceptron.py
>>> from perceptron import Perceptron
>>> pn = Perceptron(0.1, 10)
>>> pn.fit(X, y)
>>> plt.plot(range(1, len(pn.errors) + 1), pn.errors, marker='o')
>>> plt.xlabel('Epochs')
>>> plt.ylabel('Number of misclassifications')
>>> plt.show()
```

```
from matplotlib.colors import ListedColormap
def plot_decision_regions(X, y, classifier, resolution=0.02):
   # setup marker generator and color map
   markers = ('s', 'x', 'o', '^', 'v')
   colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
   cmap = ListedColormap(colors[:len(np.unique(y))])

   # plot the decision surface
   x1_min, x1_max = X[:,  0].min() - 1, X[:, 0].max() + 1
   x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
   xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
   np.arange(x2_min, x2_max, resolution))
   Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
   Z = Z.reshape(xx1.shape)
   plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
   plt.xlim(xx1.min(), xx1.max())
   plt.ylim(xx2.min(), xx2.max())

   # plot class samples
   for idx, cl in enumerate(np.unique(y)):
      plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
      alpha=0.8, c=cmap(idx),
      marker=markers[idx], label=cl)
```

Code is modified version of the code in "Python Machine Learning by Sebastian Raschka"