THE UNIVERSITY OF HONG KONG
Department of Computer Science
COMP3270B Artificial Intelligence
Additional Notes for Assignment 3

1. (a) You have to import the library that is being used in the question:

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
```

   i. The package `numpy` is the fundamental package for scientific computing in python

   ii. The package `pandas` is to read the input data (in comma-separated-values format, csv) into a data frame. The data file from UCI is in csv format.

   iii. `sklearn`, `tree`, `DecisionTreeClassifier` are for decision trees.

   iv. `preprocessing` is to convert the categorical input (in english letters) to numeric data.

   (b) Read data: (where `mushrooms` is a data frame variable, you can use any suitable name in your program)

```
mushrooms = pd.read_csv('./mushroom.csv')
```

   If you are using Google Colab (recommended), you can use:

```
from google.colab import files
uploaded = files.upload()
import io
mushrooms = pd.read_csv(io.BytesIO(uploaded['mushroom.csv']))
```

   This will prompt you for a local file to upload to Google CoLab.
   You can print out the first 5 rows to have a look:

```
print (mushrooms.head())
```

(c) Transform categorical data into numbers, because the sklearn decision tree package take numerical input, using the `LabelEncoder()` function in `preprocessing`.

For each `col` in the `mushrooms`, repeat the following:

```
for col in mushrooms.columns.values:
```

1. Define a new `LabelEncoder()` function named `le`.
   ```
   le = preprocessing.LabelEncoder()
   ```
2. the set of unique label in each attribute is given by:
   ```
   list(mushrooms[col].unique())
   ```
3. give a number for each unique label (`le_fitted` is just a dummy variable):
   ```
   le_fitted = le.fit(list(mushrooms[col].unique())
   ```
4. transform the entire column in the mapping obtained in 2.):
   ```
   mushrooms[col] = le.transform(list(mushrooms[col].values))
   ```

(d) You can print out the first 5 rows to check:

```
print (mushrooms.head())
```

(e) Prepare training and test data. `Y_train` is the class label, `X_train` is the input attributes. The `train_test_split()` function will split the data into training set and test (validation) set. The first column is the class label, hence `mushrooms.values[0:, 0]`, column 0 of all data. The remaining columns are the attribute, hence `mushrooms.values[0:, 1:23]`, column 1 to 22 of all data. Size of test data is 25% of all data, i.e 75% training data. Shuffle the data before splitting. This is needed because for some dataset, data of the same class will all be grouped together. If you get the data sequentially, those classes with data at the end of the file will not be read.

```
X_train, X_test, Y_train, Y_test = train_test_split (
   mushrooms.values[0:,1:23], mushrooms.values[0:,0],
   test_size=0.25, shuffle=True)
```

You may print out the first 5 training data to check. Note that `X_train[]` and `Y_train[]` are not data frame variables and hence cannot use `.head()`.

```
print X_train[0:5]
print Y_train[0:5]
```

(f) Now you can call the sklearn decision tree package for training:

```
clf=tree.DecisionTreeClassifier() # define a new classifier clf
clf=clf.fit(X_train, Y_train)     # perform training
```

(g) After training, you can perform testing (`Y_predict` is the predicted label):

```
Y_predict = clf.predict(X_test)
```

(h) Finally you can compare `Y_test` and `Y_predict` to obtain accuracy of the classification:

```
accuracy_score(Y_test, Y_predict)*100   # in percentage
```

Note that you can replace decision tree by other classifier easily. For example, using Naive Bayes with Gaussian distribution (though it is more suitable to numerical data):

(a) Import library:

```
from sklearn.naive_bayes import GaussianNB
```

(b) Instead of calling `tree.DecisionTreeClassifer()` call the Naive Bayes Classifier `GaussianNB()`:

```
clf=GaussianNB()
```

(c) The rest of the program remain the same.

The accuracy should be $> 90\%$.

For k-NN classifier, you can do the following:

(a) Import Library:

```
from sklearn.neighbors import KNeighbors Classifier
```

(b) to call the k-NN classifier:

```
clf=KNeighborsClassifier(n_neighbors=3)
```

(c) The rest of the program remain the same.