Part 1

Question 1. Theory(15%)

1. What is the difference between a rule-based system and a machine learning system? **(5%)**

A rule-based system is the system which rules inputted by humans (like the form of if-then-else statements) and data inputted to be processed. Comparing to a machine learning system, rule-based system just contains fixed knowledge which couldn't be changed over time. In this way a machine learning system has ability to learn therefore existing knowledge can be changed or discarded, and new knowledge can be acquired to create their own model and these models can then be applied to new data to produce original answers. Machine learning system can be regard as a real AI.

2. What is the difference between unsupervised and supervised learning? **(5%)**

In Supervised learning, we train the machine using labeled data(the data type we already know) whereas in unsupervised learning we makes use of unlabeled data(unforeseen data type). Supervised learning usually fit a function from a given training data set, and when new data arrives, we can predict the result based on this fitted function, such as Regression and Classification. But unsupervised learning has no labeled input data and there is no definite result, therefore the dataset needs to be classified based on the similarity between the different data, such as Clustering.

3. What do we mean when we say that a machine learning system is overfitting? **(5%)**

A machine learning system overfitting means this system fit a model but this model be fitted by the training data too well and the model is too complicated. This will often result in high accuracy on the training dataset, but low accuracy when applied to test data and unseen data. Therefore, new data is hardly applied by this system.

Question 2. Practice (85%)

1.Your algorithm gets the following results in a classification experiment. Please compute the precision, recall, f-measure and accuracy *manually* (without the help of your computer/Python, please provide all steps and formulas). Include the process to get to the final result. (20%)

From the table we can get the information that:
TP:6
TN:8
FP:3
FN:3

Therefore:
precision = TP/(TP+FP) = 0.67

recall = TP/(TP+FN) = 0.67

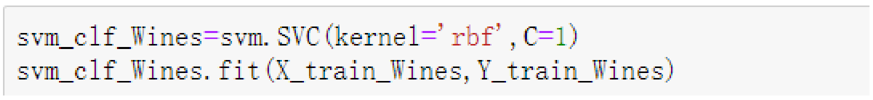
accuracy =(TP+TN)/(TP+TN+FP+FN) = 0.7

f-measure = 2*precision*recall/(precision+recall) = 0.67

2. You are given a dataset (named Wine dataset) with different measured properties of different wines (dataset available in Learning Central). Your goal is to develop a machine learning model to predict the quality of an unseen wine given these properties. Train two machine learning regression models and check their performance. Write, for each of the models, the main Python instructions to train and predict the labels (one line each, no need to include any data preprocessing) and the performance in the test set in terms of Root Mean Squared Error (RMSE) (30%)

For this unseen wine quality prediction problem, we firstly think about the SVM regression model with Gaussian kernel, because we can see the wine quality could be totally classified as 1, 2, 3, 4, 5, 6 and 7. Therefore this case is a classification problem.

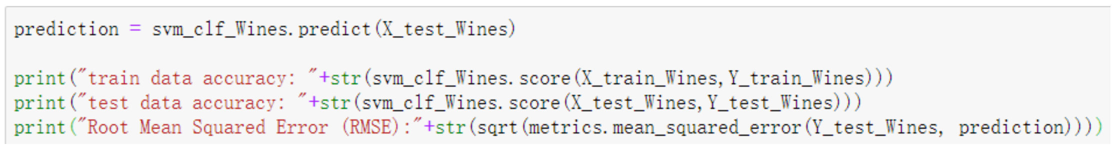
The train:



```
svm_clf_Wines=svm.SVC(kernel='rbf',C=1)
svm_clf_Wines.fit(X_train_Wines,Y_train_Wines)
```

Here the penalty parameter is 1

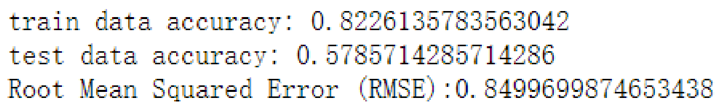
Do the predict:



```
prediction = svm_clf_Wines.predict(X_test_Wines)

print("train data accuracy: "+str(svm_clf_Wines.score(X_train_Wines,Y_train_Wines)))
print("test data accuracy: "+str(svm_clf_Wines.score(X_test_Wines,Y_test_Wines)))
print("Root Mean Squared Error (RMSE):"+str(sqrt(metrics.mean_squared_error(Y_test_Wines, prediction))))
```

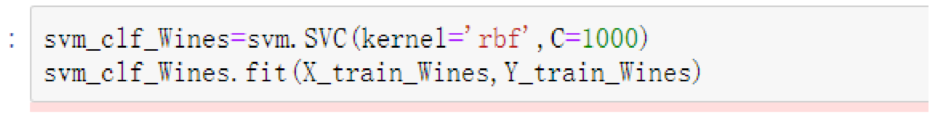
Output:



```
train data accuracy: 0.8226135783563042
test data accuracy: 0.5785714285714286
Root Mean Squared Error (RMSE):0.8499699874653438
```

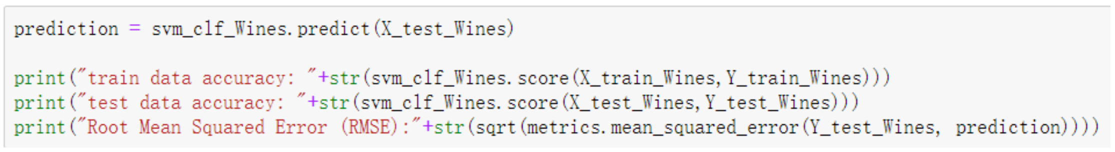
But we find that the test data accurate is really low at this moment, therefore I still using SVM regression model but here set the penalty parameter to 1000:

The code:



```
: svm_clf_Wines=svm.SVC(kernel='rbf',C=1000)
  svm_clf_Wines.fit(X_train_Wines,Y_train_Wines)
```

Do the predict:



```
prediction = svm_clf_Wines.predict(X_test_Wines)

print("train data accuracy: "+str(svm_clf_Wines.score(X_train_Wines,Y_train_Wines)))
print("test data accuracy: "+str(svm_clf_Wines.score(X_test_Wines,Y_test_Wines)))
print("Root Mean Squared Error (RMSE):"+str(sqrt(metrics.mean_squared_error(Y_test_Wines, prediction))))
```

Output:

```
train data accuracy: 1.0
test data accuracy: 0.6010204081632653
Root Mean Squared Error (RMSE):0.7693877551020408
```

For this kind of tasks, we may think the SVM is the best choice. Comparing the RMSE of both models indicates that SVM model with C=1000 is better than the model with model C=1, this maybe the reason that the model 1 hadn’t be fitted enough with a little penalty. The RMSE measures the difference between the predictions and the correct results. Therefore, the second model indicates a more accurate model, and the high accuracy output in second model also proved this.

3. Train an SVM binary classifier using the Hateval dataset (available in Learni
ng Central). The task consists of predicting whether a tweet represents hate
speech or not. You can preprocess and choose the features freely. Evaluate t
he performance of your classifier in terms of accuracy using 10-fold cross-vali
dation. Write a table with the results of the classifier (accuracy, precision, re
call and F-measure) in each of the folds and write a small summary (up to
500 words) of how you preprocessed the data, chose the feature/s, and train
ed and evaluated
your model (35%)

data preprocessed

We find that each comment in the file is the byte type not the string type. Therefore, to preprocess the data I need decode each sentence then strip them. After that I extract the review type from each comment, 0 is not hate comment ,1 is hate comment, according this I add them to different list dataset_file_hate [] ,
dataset_file_notHate[] . For easily calculate, then I combine these hate and not hate comment together again as a full set as the type of list of Tuple: dataset_full<review,reviewType>[]. Here I haven't split it to train set, test set and dev set as previously because in this project we will use 10-fold cross-validation later.

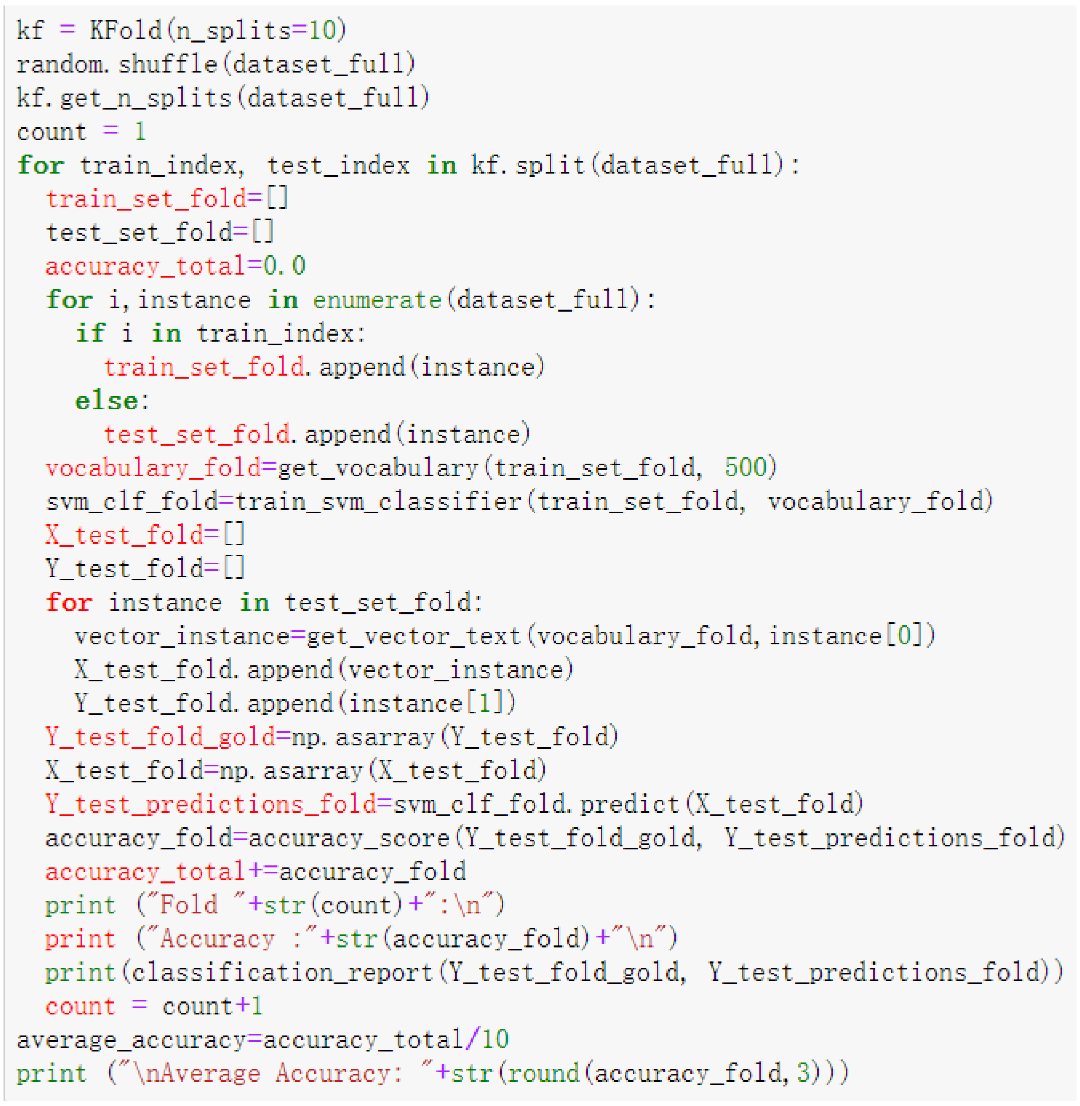
feature chosen

For this kind of comment problem, we firstly think about the SVM regression model, because we can see the comment type could be totally classified as hate and not hate. Therefore, this case is a classification problem.

Next, the feature what I choose in this problem is the frequency of each word in the text which will be used to fit the model. So firstly, I write a function to traverse each sentence and extract the words from them and append each word into a dictionary, and count the frequency of each word, then sort every word from sentence from high frequency to low frequency. At the same time, I use function from nltk module to set some stop words, in this way we can skip some punctuation mark to save time and improve efficiency. After extracting a total of top 1000 features, it is necessary to start training the data.

model trained and evaluated

In this project we used 10-fold cross validation. This is a good way to maximize the dataset on the machine learning model, and also good for training data and report outputs very intuitively. For each fold of the cross-validation the data was split into a training and test set and the attributes were calculated for that fold only. The code snip is as below:



```
kf = KFold(n_splits=10)
random.shuffle(dataset_full)
kf.get_n_splits(dataset_full)
count = 1
for train_index, test_index in kf.split(dataset_full):
  train_set_fold=[]
  test_set_fold=[]
  accuracy_total=0.0
  for i,instance in enumerate(dataset_full):
    if i in train_index:
      train_set_fold.append(instance)
    else:
      test_set_fold.append(instance)
  vocabulary_fold=get_vocabulary(train_set_fold, 500)
  svm_clf_fold=train_svm_classifier(train_set_fold, vocabulary_fold)
  X_test_fold=[]
  Y_test_fold=[]
  for instance in test_set_fold:
    vector_instance=get_vector_text(vocabulary_fold,instance[0])
    X_test_fold.append(vector_instance)
    Y_test_fold.append(instance[1])
  Y_test_fold_gold=np.asarray(Y_test_fold)
  X_test_fold=np.asarray(X_test_fold)
  Y_test_predictions_fold=svm_clf_fold.predict(X_test_fold)
  accuracy_fold=accuracy_score(Y_test_fold_gold, Y_test_predictions_fold)
  accuracy_total+=accuracy_fold
  print ("Fold "+str(count)+":\n")
  print ("Accuracy :"+str(accuracy_fold)+"\n")
  print(classification_report(Y_test_fold_gold, Y_test_predictions_fold))
  count = count+1
average_accuracy=accuracy_total/10
print ("\nAverage Accuracy: "+str(round(accuracy_fold,3)))
```

And accuracy, precision, recall, and F-measure for each fold also has been print and organize as below:

Fold	Accuracy	Precision	Recall	F-measure
1	0.77	0.77	0.77	0.77
2	0.78	0.78	0.78	0.77
3	0.76	0.76	0.76	0.75
4	0.75	0.76	0.75	0.75
5	0.76	0.76	0.76	0.76
6	0.74	0.74	0.74	0.73
7	0.78	0.78	0.78	0.78
8	0.77	0.77	0.77	0.76
9	0.79	0.79	0.79	0.79
10	0.80	0.80	0.80	0.80
Average	0.77	0.77	0.77	0.77

From the table we can clearly see the accuracy, precision, recall and F-measure of the model is around 77%. This shows that lots of comments were correctly predicted and the fitted model works very well, which can explain that the correct features and parameters are selected, and this model will be perfectly applied to the classification of most new comments.