Module Code	Examiner	Department	Tel
INT104	Shengchen Li	INT	3077

2nd SEMESTER 23-24 FINAL EXAMINATION

Undergraduate

Artificial Intelligence

TIME ALLOWED: *2 hours*

INSTRUCTIONS TO CANDIDATES

1. This is a blended open-book exam and the duration is 2 hours.
2. Total marks available are 100. This accounts for 60% of the final mark.
3. Answer all questions. Relevant and clear steps should be included in the answers.
4. Please use MCQ card delivered to answer MCQ questions. Please use answer booklet for answer other questions.
5. Only English solutions are accepted.
6. The use of calculator is allowed.
7. Besides lecture notes and hand writing notes, only books (with an ISBN) are allowed. NO DICTIONARIES.

Section 1 Multiple Choice Questions

This section of the exam contains multiple-choice questions. Each question will be followed by four options A, B, C, and D. You are required to choose ONE answer that you deem to be the most appropriate.

Section 2 Computation Questions

1. Given a dataset that consists of following points below:

$$A = (1, 2), B = (2, 0), C = (1, 0), D = (1, 1), E = (4, 0)$$

Cluster the data points by agglomerative clustering with maximum **city block** distance and draw the cluster dendrogram.

(14 Marks)

Section 3 Programming Questions

2. Assume a dataset is stored in a variable `features` where each column of `features` represents a feature and each row of `features` represents a data sample. The samples belong to a certain number of classes. A variable `labels` stores the class information of each sample as a column vector where each row of `labels` represents a data sample.

The Python script on the next page attempts to calculate the ratio between **average intra-class distance** and **average inter-class distance** per pair of samples.

Both `features` and `labels` are an `ndarray` in Numpy.

Please fill in the blank marked as `[#001]` to `[#006]` as appropriate in the script and then answer the following question:

- Considering the difficulties of classification process, will a larger value of the ratio between **average intra-class distance** and **average inter-class distance** be preferred?

Each blank in the Python script is worth 2 marks. The question you are asked to answer is worth 4 marks.

A set of API of Python has been provided in the section of Appendix for your reference.

```
import numpy as np

intra_distances = []
inter_distances = []

for i in range([#001]):
    for j in range([#002], len(features)):
        # Euclidean distance between two samples
        distance = np.linalg.norm([#003])

        # If both samples belong to the same class,
        # it’s an intra-class distance
        if [#004]:
            intra_distances.append(distance)
        # Else, it’s a inter-class distance
        else:
            inter_distances.append([#005])

# Calculate average intra-class and inter-class distances
avg_intra_distance = np.mean(intra_distances)
avg_inter_distance = np.mean(inter_distances)

# Calculate the ratio
ratio = [#006]

print(‘Ratio:’, ratio)
```

(16 Marks)

3. Assume a dataset is stored in a variable `features` where each column of `features` represents a feature and each row of `features` represents a data sample. The samples belong to **4 classes**. A variable `labels` stores the class information of each sample as a column vector where each row of `labels` represents a data sample.

The Python script on next page attempts to build up an ensemble classifier that predict the class of samples belonging to. Three sub-classifiers are built: a SVM classifier, a decision tree classifier and a kNN classifier. The final prediction is made by a vote of three sub-classifiers.

Both `features` and `labels` are an `ndarray` in Numpy.

Please fill in the blank marked as `[#007]` to `[#012]` as appropriate in the script and then answer the following question:

- Besides voting, propose two other methods to make the final prediction with prediction of sub-classifiers.

Each blank in the Python script is worth 2 marks. The question you are asked to answer is worth 4 marks.

A set of API of Python has been provided in the section of Appendix for your reference.

```
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split \
    (features, labels, test_size=0.2)

# Create the classifiers
knn = KNeighborsClassifier(n_neighbors=5)
svm = SVC(kernel='rbf', probability=True, random_state=104)
dt = DecisionTreeClassifier(random_state=104)

# Create the ensemble model
ensemble = VotingClassifier(estimators=[('knn', knn), \
    ('svm', svm), ('dt', dt)], voting='hard')

# Train the ensemble model
[#007].fit([#008], [#009])

# Check the ensemble model performance
ensemble_score = ensemble.[#010]([#011], [#012])

print('Ensemble Model Accuracy:', ensemble_score)
```

(16 Marks)

Section 4 Appendix: Edited Python API being used in this exam

The following API information may be used in this exam.

numpy.linalg.norm

`linalg.norm(x)`: Matrix or vector norm.

Parameters

`x: array_like`

Input array. If axis is None, x must be 1-D or 2-D, unless ord is None.

Returns

`n: float or ndarray`

Norm of the matrix or vector(s).

numpy.mean

`numpy.mean(a)`: Compute the arithmetic mean.

Parameters

`a: array_like`

Array containing numbers whose mean is desired. If a is not an array, a conversion is attempted.

append() in `list`

`list.append(x)`: Add an item to the end of the list.

Equivalent to `a[len(a):] = [x]`.

sklearn.model_selection.train_test_split

`sklearn.model_selection.train_test_split(*arrays, test_size=None)`: Split arrays or matrices into random train and test subsets.

Parameters

`*arrays: sequence of indexables with same length / shape[0]`

Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.

`test_size: float or int, default=None`

If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If int, represents the absolute number of test samples.

Returns

`splitting: list, length=2 * len(arrays)`

List containing train-test split of inputs.

sklearn.ensemble.VotingClassifier

`class sklearn.ensemble.VotingClassifier(estimators, *, voting='hard'):`
Soft Voting/Majority Rule classifier for unfitted estimators.

Parameters

`estimators: list of (str, estimator) tuples`

Invoking the fit method on the VotingClassifier will fit clones of those original estimators that will be stored in the class attribute self.estimators_.

`voting: 'hard', 'soft', default='hard'`

If 'hard', uses predicted class labels for majority rule voting. Else if 'soft', predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

Methods

`fit(X, y)`

Fit the estimators.

<u>Parameters</u>

`X: array-like, sparse matrix of shape (n_samples, n_features)`

Training vectors, where n_samples is the number of samples and n_features is the number of features.

`y: array-like of shape (n_samples,)`

Target values.

<u>Returns</u>

`self: object`

Returns the instance itself.

`score(X, y)`

Return the mean accuracy on the given test data and labels.

In multi-label classification, this is the subset accuracy which is a harsh metric since you require for each sample that each label set be correctly predicted.

Parameters

`X: array-like of shape (n_samples, n_features)`

Test samples.

`yarray-like of shape (n_samples,)`

True labels for X.

Returns

`score:  float`

Mean accuracy of self.predict(X) with reference to y.

END OF EXAM PAPER
THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM