Introduction to Machine Learning (Spring 2019)

Homework #6 (50 Pts, June 23)

Student ID 2014310407

Name 이 준 혁

Instruction: We provide all codes and datasets in zip files. Please write your code to complete Decision Tree and Gaussian Naïve Bayesian models. Compress only 4 Files 'Decision_Tree_Answer.py', 'NaïveBayesian_Answer.py' 'utils_Answer.py' & your report and submit with the filename 'HW6_STUDENT_ID.zip'.

1 – (1) [10 pts] Implement Decision Tree in 'Decision_Tree_Answer.py', some parts of 'utils_Answer.py'

$$Gini = \sum_{i=1}^{m} p_i(1 - p_i),$$

$$Entropy = -\sum_{i=1}^{k} p_i log_2 p_i,$$

where N is the number of (batch) data, C is the number of classes.

(a) [Gini index] Implement Gini_index in 'utils_Answer.py'

```
def Gini_index(Y_data):
    gini = 0
    #=========    Edit here    ==========
    if len(Y_data) != 0:
        _, counts = np.unique(Y_data, return_counts= True)
        counts = np.divide(counts, sum(counts))
        gini = 1 - np.sum(np.multiply(counts, counts))
    #==================================
    return gini
```

(b) [Entropy] Implement Entropy in 'utils_Answer.py'

```
def Entropy(Y_data):
    entropy = 0
    # =====    Edit here    ========
    if len(Y_data) != 0:
        _, counts = np.unique(Y_data, return_counts=True)
        counts = np.divide(counts, sum(counts))
        entropy = -np.sum(np.multiply(counts, np.log2(counts)))
    # ============================
    return entropy
```

(c) [Find Best feature] Implement a 'Find_Best_Feature' in 'Answer.py'.

```
def Find_Best_Feature(self, df):
    header = df.columns.values
    input_feature = header[:-1]
    output_feature = header[-1]
    Y_data = df[output_feature].values

    impurity_list = []

    # for all features in DataFrame,
    for idx, h in enumerate(input_feature):
    # ===========       Edit here      =================
        # Category Feature Case
        if idx in self.Category_feature_idx:
            impurity = 0
            feature = input_feature[idx]
            X_data = df[feature].values
            unique = np.unique(X_data, return_counts=False)

            for feature_value in unique:
                splited_Y_data = Y_data[feature_value == X_data]
                impurity += impurity_func(splited_Y_data, self.criterion) * len(splited_Y_data)
            impurity /= len(Y_data)

        # Numeric Feature Case
        else:
            split_value = Finding_split_point(df, h, self.criterion)
            impurity = 0

            feature = input_feature[idx]
            X_data = df[feature].values

            splited_Y_data_large = Y_data[X_data >= split_value]
            splited_Y_data_less = Y_data[X_data < split_value]

            impurity += impurity_func(splited_Y_data_large, self.criterion) *
                                len(splited_Y_data_large)
            impurity += impurity_func(splited_Y_data_less, self.criterion) *
                                len(splited_Y_data_less)

            impurity /= len(Y_data)

    #================================================
        impurity_list.append(np.round(impurity, 6))

    idx = np.argmin(impurity_list)

    Best_feature = input_feature[idx]
    feature_type = idx in self.Category_feature_idx and 'Category' or 'Numeric'

    return Best_feature, feature_type
```

1 – (2) [10 Pts] Experiment results

(a) you are given 2 dataset (Heart, Carseats) with Binary classification(Yes or No). Measure the performance of Decision tree given setting environments.

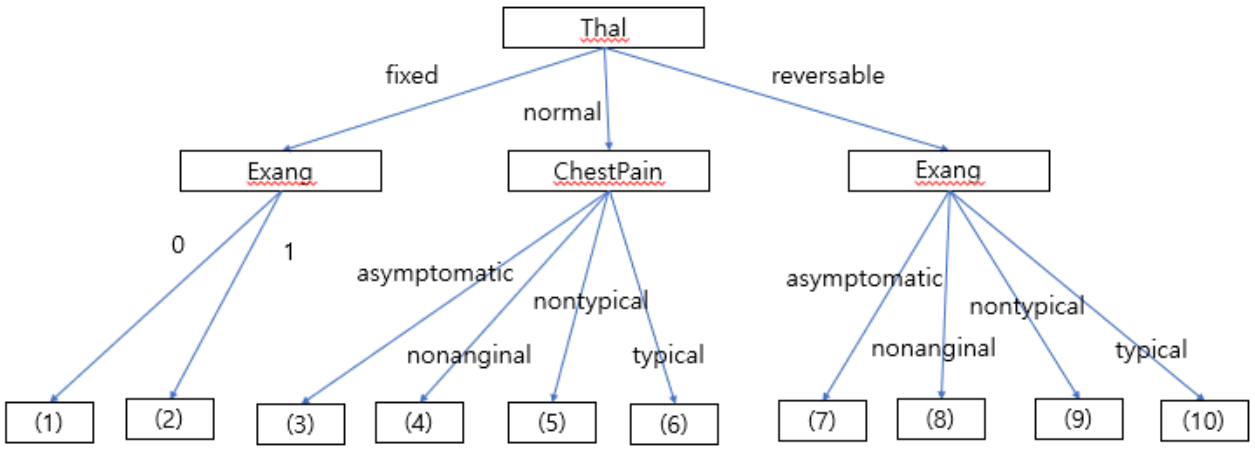
Answer: Fill the blank in the table. Show the plot of training & test accuracy with a brief explanation.

[Decision Tree]

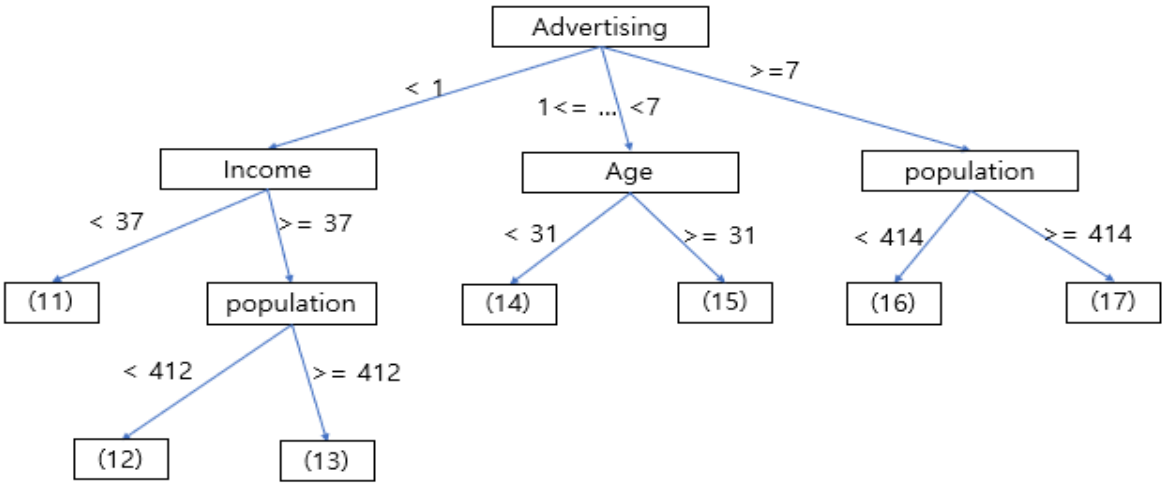
Dataset	Impurity function	Max depth	Accuracy
Heart	Entropy	2	0.7368
Heart	Gini_index	3	0.7368
Carseats	Entropy	3	0.9125
Carseats	Gini_index	3	0.8875

1 – (3). [10 pts] Analysis

Heart_dataset, (Impurity function = entropy, max_depth = 2)



Carseat_dataset, (Impurity function = entropy, max_depth = 3)



Above figure is the result of Decision Tree for the given condition. Please fill the label of leaf node. And Explain which node can be pruned.

(1) No	(2) Yes	(3) No	(4) No	(5) No
(6) No	(7) Yes	(8) Yes	(9) No	(10) No
(11) No	(12) No	(13) No	(14) No	(15) Yes
(16) Yes	(17) Yes			

1번 트리:

1) Thal이 normal인 경우는 다 No이므로, prune할 수 있다.

2번 트리:

1) Advertising이 1보다 작은 경우는 다 No이므로, prune할 수 있다.

2) Advertising이 7보다 큰 경우는 다 Yes이므로, prune할 수 있다.

2 - (1). **[10 pts]** Implement Gaussian Naïve Bayesian in 'NaiveBayesian_Answer.py' and the rest parts of 'utils _Answer.py' (To prevent that likelihood becomes zero, please apply the Laplacian smoothing technique.).

(a) **[likelihood, prior]** Implement 'fit' in 'NaïveBayesian_Answer.py' for the following function: (likelihood, prior).

```
# P(Yes), P(No)
  # ======     Edit this     ======
    self.prob_yes = len(pos_data) / (len(pos_data) + len(neg_data))
    self.prob_no = len(neg_data) / (len(pos_data) + len(neg_data))
  # ============================
```

(b) **[Posterior]** Implement 'predict' in 'NaiveBayesian_Answer.py' for the posterior probability.

```
def get_posterior(self, tuple, likelihood, prior):
    feature = tuple.index
    posterior = 1

    # ===========           Edit here             ==============
    for idx, f in enumerate(feature):
        val = tuple[f]

        if idx in self.Category_feature_idx:
          # to avoid key Error in predict fuction.
            if type(val) == np.float64 or type(val) == np.float32:
                val = int(val)
            posterior *= likelihood['%s = %s' %(f, val)]

        else:
            mean = likelihood['%s_mean' % (f)]
            std = likelihood['%s_std' % (f)]
            posterior *= Gaussian_prob(val, mean= mean, std= std)

    posterior *= prior
    # ============================================================
    return posterior
```

(c) **[Gaussian probability]** Implement Gaussian_prob in 'utils_Answer.py'. for the following function:

```
def Gaussian_prob(x, mean, std):
    ret = 0
    # ========      Edit here         ==========
    ret = np.exp(- (((x - mean) / std) ** 2) / 2) / (np.sqrt(2 * np.pi) * std)
    # ======================================
    return ret
```

2 - (2). **[10 pts]** Experimental results.

(a) you are given 2 dataset (Heart, Carseats) with Binary classification(Yes or No). Measure the performance of Decision tree and fill the blank.

[Naïve Bayesian]

Dataset	Accuracy
Heart	0.7719
Carseats	0.9