CS 3753 & 5163 Data Science

Homework 5 (100 +20 points)

Questions

1. (20 points) We do the linear regression on three points (0.5, 1), (2, 2.5), and (3, 3). Please calculate the SSEs of the four following linear regression based on the definition $SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. Write the Python code to output the major steps and results.
   (a) y = x + 0.5
   (b) y = x + 1
   (c) y = 0.8*x + 0.3
   (d) y = 0.8*x + 0.7
   Which is the best linear regression using SSE?

2. (20 points) What is the problem solved by Lasso and Ridge regression? What is the major difference between the two regression? Please discuss the advantages and disadvantages of them.

3. (30 pints) Decision Tree
   There are various ways to decide on the metric to choose the variable on which splitting for a node is done. Different algorithms deploy different metrices to decide which variable splits the dataset best.

   Let's say we have a sample of 30 records. There are two classes C1 and C2. We have three possible splits a, b, and c (see figure below). The number of records in each class is shown in every node.

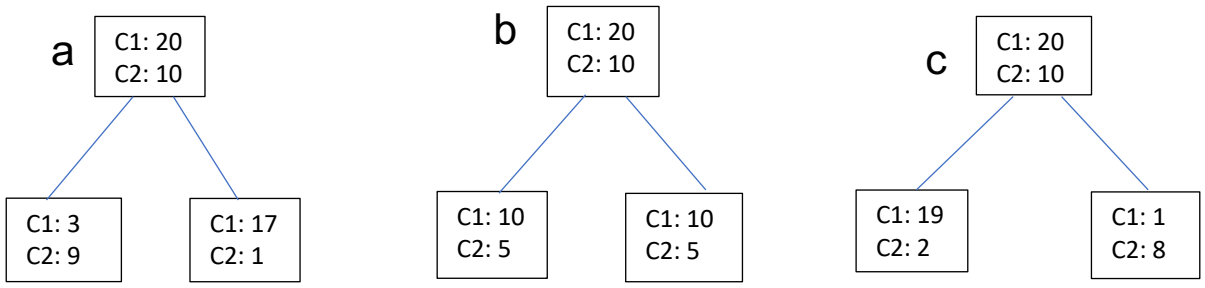
   a. Write a Python code to measure the node impurity using Gini Index, Entropy Gain, and Misclassification Error, respectively. The following tables will help you understand the question. Please output the results in your Python code.
   The output information of node impurities is in the following table

	a		b		c	
Impurity	left	right	left	right	left	right
Gini index						
Entropy						
Misclass error						

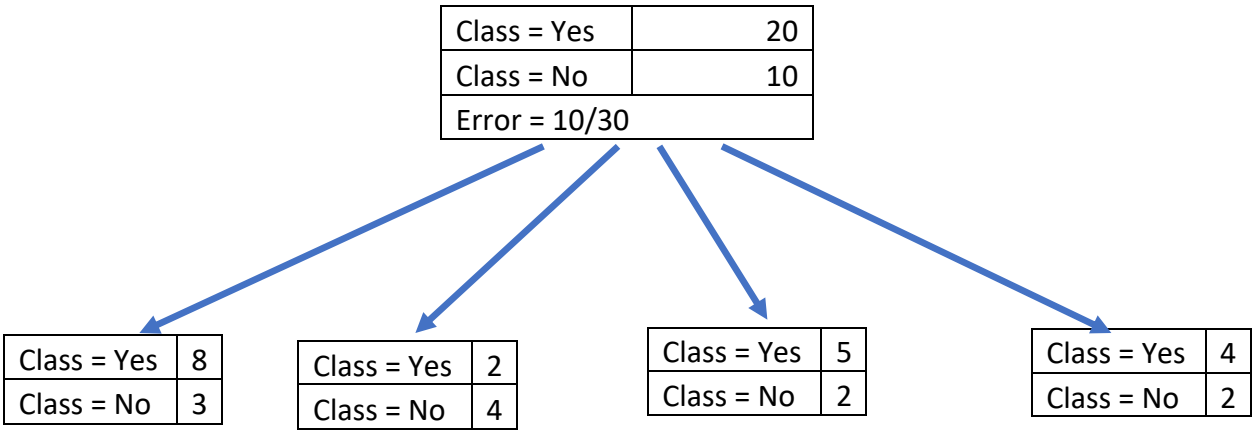
   b. Evaluate the quality of the three splitting and report the best one.
   The output qualities of the splitting are in the following table

Splitting	a	b	c
Gini index			
Entropy			
Misclass error			

c. The code should print out the quality of each splitting and your best choice. You also can print out any other information freely.
d. Finally, print out your conclusion about whether all three methods have the same best choice.



4. (**Extra credits:** 20 pts) Post pruning of decision tree by pessimistic approach. In this approach, the error in a leaf node is $e'(t) = e(t) + 0.5$, where $e(t)$ is the training error in a node. Please calculate the pessimistic error at the parent node A and leaf nodes. Should this tree be pruned?



5. (30 points) KNN: this section applies the KNN algorithm to the Iris flowers dataset. The first step is to load the dataset in "iris.csv" and convert the loaded data to numbers that we can use with the mean and standard deviation calculations. For this we will use the helper function load_csv() to load the file, str_column_to_float() to convert string numbers to floats and str_column_to_int() to convert the class column to integer values.

You do not need to import any libraries or modules about KNN because you will implement the KNN from scratch. The template of the code is provided and you just need to complete the functions Euclidean_distance(), get_neighbors(), and predict_classification(). The mean accuracy is around 96.667% (≈97%).

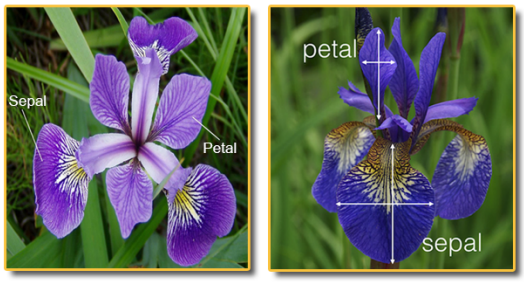
We will evaluate the algorithm using k-fold cross-validation with 5 folds. This means that 150/5=30 records will be in each fold. We will use the helper functions evaluate_algorithm() to evaluate the algorithm with cross-validation and accuracy_metric() to calculate the accuracy of predictions.

A new function named k_nearest_neighbors() was developed to manage the application of the KNN algorithm, first learning the statistics from a training dataset and using them to make predictions for a test dataset.

Download the dataset and save it into your current working directory with the filename "iris.csv". The Iris Flower Dataset involves predicting the flower species given measurements of iris flowers.

It is a multiclass classification problem. The number of observations for each class is balanced. There are 150 observations with 4 input variables and 1 output variable. The variable names are as follows:

a. Sepal length in cm.
b. Sepal width in cm.
c. Petal length in cm.
d. Petal width in cm.
e. Class



1.

Python code:

```
# importing numpy array to to vectorize thing

import numpy as np

# Since we have three points to fit

# Lets define list x_data and y_data for points

x_data = np.array([ 0.5 , 2 , 3 ])

y_data = np.array([ 1 , 2.5 , 3 ])

# Lets get y_predicted value for all four models

y_predicted = []

# Defining parameter of models

model = [(1 , 0.5) , (1 , 1) , (0.8 , 0.3) , (0.8 , 0.7)]

for model_i in model:

(a , b) = model_i

y = []

for x in x_data:

yi = a * x + b

y.append(yi)

y_predicted.append(y)

y_predicted = np.array(y_predicted)

# Now we will calculate error for each model

error = []

for y_pred in y_predicted:

error.append(np.sum(np.power(y_data - y_pred , 2)) / len(y_pred))

# Displaying error obtained from each model

for i in range(0 , len(error)):

print("Error obtained from model {} is {} ".format(i , error[i]))
```

2.

If your modeling problem is that you have too many features, a solution to this problem is LASSO regularization. By forcing some feature coefficients to be zero, you remove them, thus reducing the number of features that you are using in your model. LASSO solves the problem of too many features through feature selection. Ridge regression is primarily a tool for dealing with collinearity. It does this by allowing some bias in exchange for greatly reducing the variance of the estimators.

Difference

1. Lasso and Ridge are all part of the Linear Regression family where the x (input) and y (output) are assumed to have a linear relationship. The main difference among them is whether the model is penalized for its weights.

2. Lasso is a modification of linear regression, where the model is penalized for the sum of absolute values of the weights. During training, the objective function become:

$(1/2m)\ \Sigma(y- Xw)^2 + \text{alpha}\ \Sigma|W_j|$

As you see, Lasso introduced a new hyper parameter, alpha, the coefficient to penalize weights.

3. Ridge takes a step further and penalizes the model for the sum of squared value of the weights. Thus, the weights not only tend to have smaller absolute values and more evenly distributed, but also tend to be close to zeros. The objective function becomes:

$\Sigma(y- Xw)^2 + \text{alpha}\ \Sigma\ W_j^{\,2}$

4. Lasso tends to give sparse weights (most zeros), because the l1 regularization cares equally about driving down big weights to small weights, or driving small weights to zeros. If you have a lot of predictors (features), and you suspect that not all of them are that important, Lasso may be really good idea to start with.

5. Ridge tends to give small but well distributed weights, because the l2 regularization cares more about driving big weight to small weights, instead of driving small weights to zeros. If you only have a few predictors, and you are confident that all of them should be really relevant for predictions, try Ridge as a good regularized linear regression method.

Advantage of Lasso regression

There are many advantages in using LASSO method, first of all it can pro- vide a very good prediction accuracy, because shrinking and removing the coefficients can reduce variance without a substantial increase of the bias, this is especially useful when you have a small number of observation

Disadvantage of LASSO:

LASSO selects at most n variables before it saturates. LASSO can not do group selection. If there is a group of variables among which the pairwise correlations are very high, then the LASSO tends to arbitrarily select only one variable from the group

Advantage of ridge regression

Advantage is coefficient shrinkage and reducing model complexity. The most important one is that it penalizes the estimates. Adding a penalty term reduces overfitting. Second, the penalty term guarantees that we can find a solution.

Disadvantage of ridge regression

It will shrink the coefficients for least important predictors, very close to zero. But it will never make them exactly zero. In other words, the final model will include all predictors. However, in the case of the lasso, the L1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.