Problem Set#1

Asif Rehan, arehan7@gatech.edu

Problem#4

Part 1: Explain how you can use Decision Trees to perform regression?

Decision Trees can be used to perform regression by choosing the node split criteria which is suitable for regression. Instead to using Information Gain or Entropy, replacing Information Gain with Standard Deviation Reduction can help in this regard. Using Mean Squared Error reduction as the criterion, this can reduce the loss. Also the leaf values are numeric values for regression instead of classes as in classification problems. So the final leaf regression values are taken to be some central values of the target variable such as mean or median etc. instead of the maximum voting class. With these basic modifications Decision Trees can be used to perform regression.

Part2: Show that when the error function is squared error, then the expected value at any leaf is the mean.

When the error function is Squared Error, then the function form is:

$$SE(w) = E[(Y-w)^2] = \int_{-\infty}^{+\infty} (y - w)^2 p(y)\, dy; \quad since,\ E[Y] = \int_{-\infty}^{+\infty} y\, p(y)\, dy$$

and weight w is a constant and p(y) is the probability distribution function for the distribution over target variable Y

So,

$$SE(w) = E[(Y-w)^2] = \int_{-\infty}^{+\infty} y^2 p(y) dy - 2w \int_{-\infty}^{+\infty} yp(y)\, dy + w^2 \int_{-\infty}^{+\infty} dy = \int_{-\infty}^{+\infty} y^2 p(y) dy - 2w \int_{-\infty}^{+\infty} yp(y)\, dy + w^2$$

since $\int_{-\infty}^{+\infty} dy = 1$ is the sum of over the distribution

Finally, to optimize the SE(w) function w.r.t w and setting it to 0,

$$0 = 0 - 2\int_{-\infty}^{+\infty} yp(y) dy + 2w$$

$$\Rightarrow w = \int_{-\infty}^{+\infty} yp(y) dy = E[Y]$$ that means, **when the error function is squared error, then the expected value at any leaf is the mean** [1].

Part 3: Take the Boston Housing dataset (https://archive.ics.uci.edu/ml/datasets/Housing) and use Decision Trees to perform regression.

The Boston Housing data is loaded in Python from Scikit-Learn as it was not available in UCI website. The data had 14 variables. There are 14 attributes in each case of the dataset. Where features are:

```
CRIM - per capita crime rate by town
ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS - proportion of non-retail business acres per town.
CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
NOX - nitric oxides concentration (parts per 10 million)
RM - average number of rooms per dwelling
AGE - proportion of owner-occupied units built prior to 1940
DIS - weighted distances to five Boston employment centres
RAD - index of accessibility to radial highways
TAX - full-value property-tax rate per $10,000
PTRATIO - pupil-teacher ratio by town
B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT - % lower status of the population
```

Target Variable:

```
MEDV - Median value of owner-occupied homes in $1000's
```

It was split into 75% train and 25% test dataset. Then min samples to split, min samples for leaf, depth of tree and pruning parameter CCP_alpha were put for GridSearch using 5fold cross validation on the training data.

```
X, y = load_boston(return_X_y=True)

 X_train, X_test, y_train, y_test = train_test_split(X, y)

 pipe = Pipeline([('std', StandardScaler()), ('cfr', DecisionTreeRegressor(random_state=42))])

 param_grid = {'cfr__min_samples_split': np.arange(10, 100, 20),
               "cfr__min_samples_leaf": np.arange(5, 100, 10),
               "cfr__max_depth": np.arange(2, 12, 5),
               "cfr__ccp_alpha": np.linspace(0, 0.1, 5),
              }


 with warnings.catch_warnings():
     warnings.filterwarnings("ignore", category=ConvergenceWarning,
                                module="sklearn")
     tuned_model = GridSearchCV(pipe, param_grid, cv=5, n_jobs=-1, scoring='r2')
     tuned_model.fit(X_train, y_train)
 print("Tuned params: {}".format(tuned_model.best_params_))

 y_pred_train = tuned_model.predict(X_train)
 y_pred_test = tuned_model.predict(X_test)
```

```
    print("Train RMSE = ", rmse(y_pred_train, y_train))
    print("Test RMSE = ", rmse(y_pred_test, y_test))

    print("Train RSquared = ", r2_score(y_pred_train, y_train))
    print("Test RSquared = ", r2_score(y_pred_test, y_test))
```

```
Tuned params: {'cfr__ccp_alpha': 0.05, 'cfr__max_depth': 7, 'cfr__min_samples_leaf': 15,
'cfr__min_samples_split': 10}
Train RMSE =  3.6916421788544027
Test RMSE =  4.2646503256680015
Train RSquared =  0.8113520328420283
Test RSquared =  0.7680431352397221
```

The tuned model has ccp_alpha=0.05, max depth of 7, requires minimum 15 samples for a leaf node, and minimum 10 samples to split a node. The training RMSE is 3.69, testing RMSE is 4.26, R-squared value was very good, 0.81 for training and 0.77 for testing. So the model did not overfit significantly and performed well for regression.

Problem#5

Suggest a lazy version of the eager decision tree learning algorithm ID3. What are the advantages and disadvantages of your lazy algorithm compared to the original eager algorithm?

A simple lazy decision tree version of ID3 does not have a trained decision tree. The idea of a lazy learner is to defer learning until an input instance is given to predict its class. Similar to a k-Nearest Neighbor algorithm, the training data can be split into a subset which matches a particular attribute value of the test instance. The subset forms a new sub-tree. This way, the resulting tree searches through the data similar to the test instance and finds the labels for the similar samples to determine the final label for the instance. When all data in the leaf have same label or same attribute values, the tree is recursively built.

A similar idea is found in the pseudocode from Friedman et al. [2] for Lazy Decision Tree

♦ Input: training set T of labelled instances. Instance I to classify.

♦ Output:a label for instance I. 1.

Algorithm:

1. If T is pure (all instances have same label L), return label L.
2. If all instances have the same feature values, return the majority class in T.
3. Select a test X and let x be the value of the test on instance I. Assign the test of instances with X=x to T and apply the algorithm to T.

Advantage:

1. Faster Training: Since no tree is being built upfront, there is no training time required!! This saves time!
2. Replication and fragmentation: As a tree is built, the number of instances in every node decreases. If many features are relevant, we may not have enough data to make the number of splits needed.
3. Typically a Decision Tree is already built for a given sample data. It builds the tree in advance. Many of the branches of this tree might be redundant for a particular instance. This leads to unnecessary traversal through. Instead a lazy Decision Tree may be shorter in height.
4. Unknown values: For an instance, if data has unknown values it is hard to handle in a prebuilt tree, but in a lazy DT, similar data can be quickly found without requiring special data handling.
5. A decision tree node keeps splitting data and eventually only a small set of data is available to make a decision. But with lazy DT, the data split based on the test instance, so a lot more data is available to make better classification decision.

Disadvantage:

1. Lazy DT lacks regularization and pruning.
2. LAzy DT would be very slow since a lot of processing is done during testing and no training is done. Some caching, and discretization of data could help.
3. Compare dynamic complexity (Holte), i.e., the number of splits until a decision is made.

Problem#7

Give the VC dimension of the following hypothesis spaces. Briefly explain your answers.

1. An origin-centered circle (2D)

N points can be classified in 2^N ways in a binary classification. In a circle function, if a point is within the circle it can be classified -1, outside +1, on the circle can be classified as 0.

An origin-centered circle (2D) can shatter any 1 point with radius r_1. If radius of circle $\mathbf{r} < r_1$, then the class could be -1, if $r > r_1$ it can classified as +1.

It can also shatter any 2 points, with radius $r_1 \leq r_2$. If $r \leq r_1 \leq r_2$, then, both points are +1. If both points are inside cicle they both classify as -1. Given $r_1 \leq r \leq r_2$, then first point is -1, and second point is +1. Again the reverse is also true. So it can shatter any 2 points.

But a circle cannot shatter three points with $r_1 \leq r_2 \leq r_3$. It cannot classify the first and third in one way and label the middle one differently. So it cannot shatter any 3 points.

So the VC dimension is 2. Reference in [3].

2. An origin-centered sphere (3D)

A sphere is a 3D version of a circle. So apparently the same logic applies as for circles. It can shatter 1 and 2 points but cannot shatter every 3 points. So the VC dimension is 2 again [3].

Reference:

1. https://www.dcc.fc.up.pt/~ltorgo/PhD/th3.pdf

2. Friedman et al. http://robotics.stanford.edu/users/ronnyk/lazyDT.pdf

3. https://alliance.seas.upenn.edu/~cis520/dynamic/2017/wiki/index.php?n=Recitations.VCDim#toc3