Lab - 2

PRML
AY 2020-21, Trimester - III

March 15, 2021 **J Jahnavi - B19CSE109**

## DecisionTreeClassifier

1) Preprocessing the data:

- Imported Libraries
- Imported the Datasets (df)
    - Extracted independent variable (x)
    - Extracted independent/target variable (y)
- The given dataset did not have any missing data
- Encoded Categorical data :

```
df = df.replace({'Yes':1
,'No':0,'High':1,'Low':0,'Morning':0,'Afternoon':1,'Evening':2,
'Hot':2,'Warm':1,'Cool':0})
```

- Splitting the Dataset into the Training set and Test set by taking two for testing and rest for training after shuffling the data

2) Cross-validation over the data:

```
s = cross_val_score(clf_gini, x, y, cv=5)
s
```

Output :

```
array([0.66666667, 1.        , 1.        , 0.33333333, 0.5       ])
```

- `cross_val_score` splits the data into 5 folds (as cv=5 for 5 fold cross-validation). Then for each fold, it fits the data on 4 folds and scores the 5th fold.
- `cross_val_score` estimates the expected accuracy of the model on out-of-training data (pulled from the same underlying process as the training data). The benefit is that one need not set aside any data to obtain this metric, and can still train the model on all of the available data. So for the given data, **the output represents the expected accuracy for each fold.**

3)Training the final model after cross-validation:

```
clf_gini = DecisionTreeClassifier(criterion='gini') #or criterion =
'entropy'
clf_gini.fit(x_train,y_train)
```

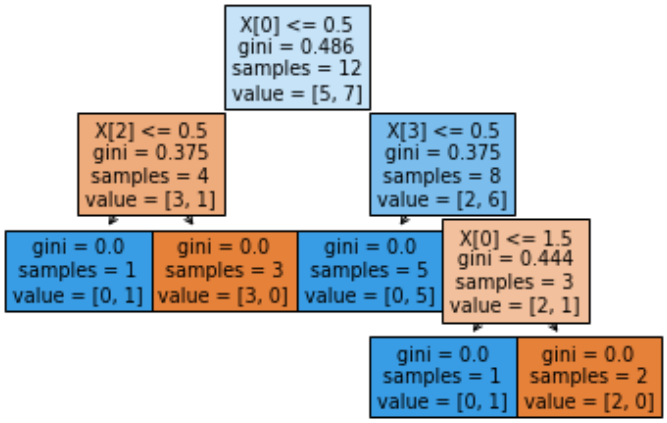
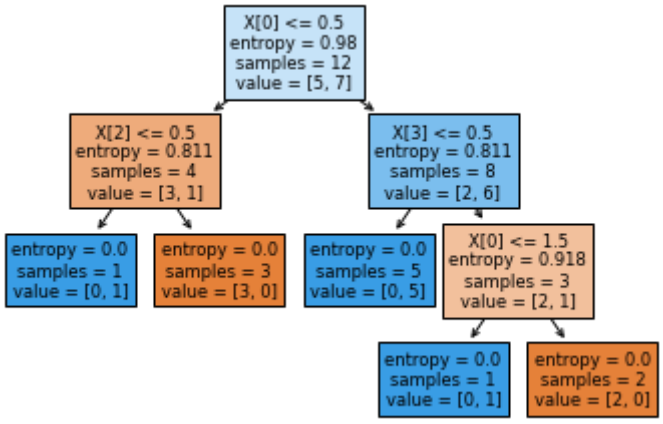
The model is now trained with the training set.

4)Accuracies:

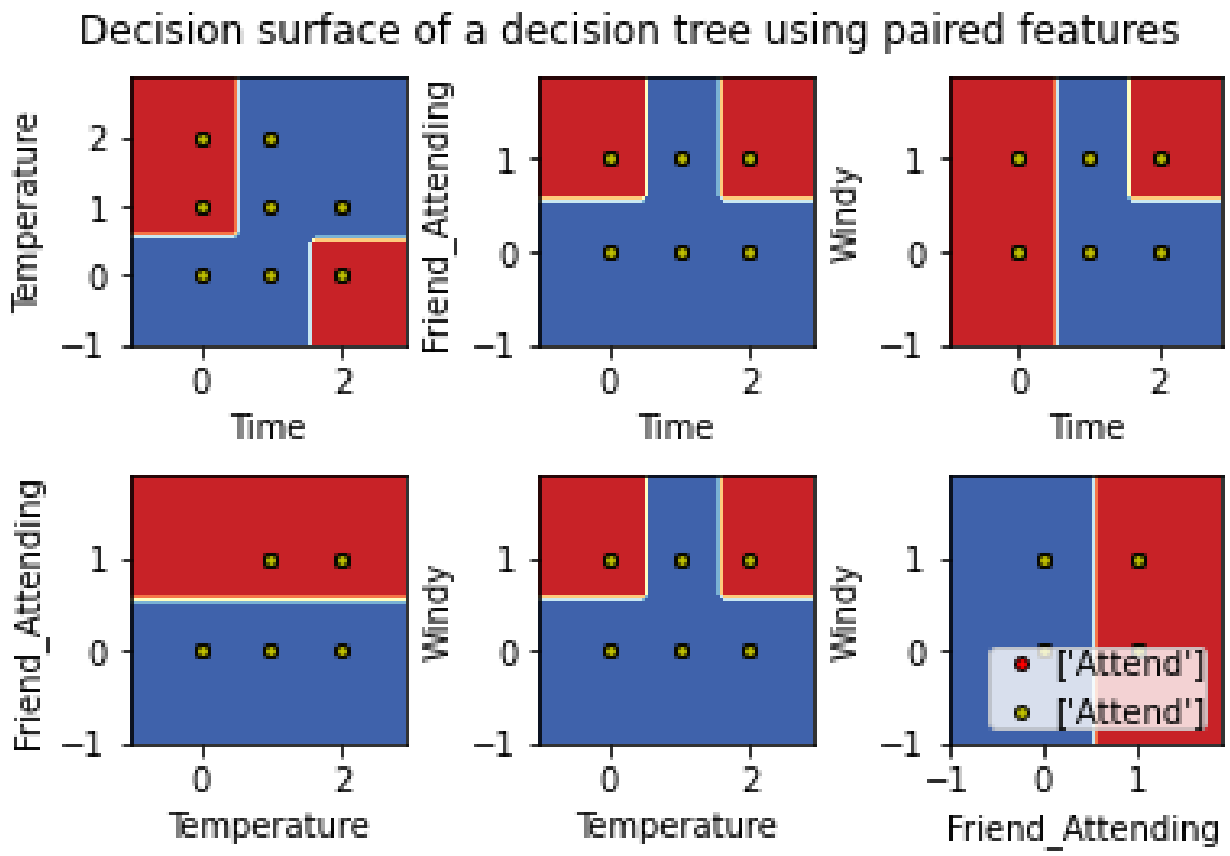
- Training Accuracy: having a small dataset with a higher train split, the training accuracy was found to be 1.0 which indicates that the model is closely fit which results in overfitting.
- Testing accuracy: the testing accuracy was found to be 1.0 which indicates that the model has predicted correctly, this might be because of having a lower test split when compared to the training set.

5)Decision tree:

Both the criterion result in a similar decision tree.

| Criterion | Decision Tree |
|---|---|
| Gini Impurity |   |
| Entropy |   |

Decision Surface:




```
DecisionTreeRegressor
```

1) Preprocessing the data:

- Imported the Datasets (dt)
  - Extracted independent variable (x1)
  - Extracted independent/target variable (y1)
- The given dataset has missing data:
  - Instead of an empty column value in a row, either '*' or '**' has been used to represent missing data
  - The method used to handle missing data: Deleted the particular row with missing data

```
dt = dt.replace({'**':'' ,'*':''})
dt.replace('',nm.nan,inplace=True)
dt.dropna(inplace=True)
```

  - Here,
    - Replaced '**' and '*' with empty strings and then dropped all the column values after replacing these empty strings with data type NaN.

- Encoded Categorical data :

```
dt = dt.replace({'stable':1, 'falling':0,'rising':2,'Yes':1,'No':0})
```

- The column ‘Average Deaths per Year’ has numeric values as type - string. So, formatted the data of this column accordingly.
- Splitting the Dataset into the Training set and Test set by taking test_size= 0.2

2) Cross-validation over the data:

```
regr = DecisionTreeRegressor(max_depth=12)
s1 = cross_val_score(regr, x1, y1, cv=5)
s1
```

Output :

```
    array([0.64898167, 0.84668175, 0.46610293, 0.08893629, 0.93326912])])
```

- For the given data, **the output represents the expected accuracy for each fold.**

3)Training the final model after cross-validation:

```
regr.fit(x1_train, y1_train)
```

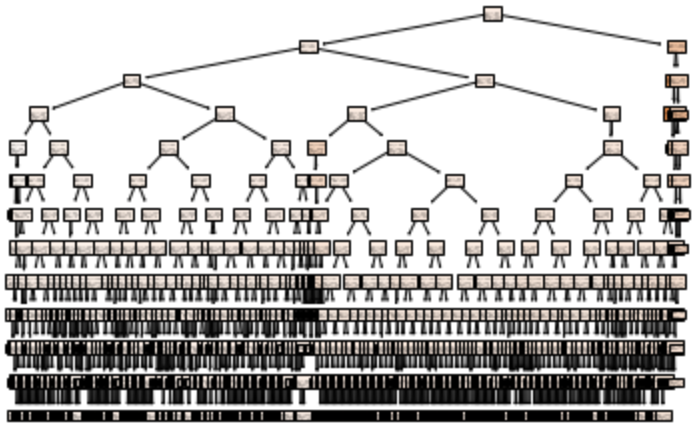
The model is now trained with the training set.

4)Squared error between the predicted and the actual values for the test data:

```
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y1_test, y1_pred))
>>> 4.716168441322922
```

- Mean squared error measures the average of the squares of the errors i.e., the average squared difference between the predicted values and the actual value.
- It was observed that **upon increasing the maximum possible depth (max_depth) for the decision tree, the mean square error was decreasing** as the prediction accuracy of the model increases.
- The better the maximum possible depth of the decision tree, the better the prediction accuracy to an extent (without close fit on the training data).

5) Decision Tree:
Model = Decision Tree Regressor
max_depth = 12



A clearer picture of the same could be displayed (included in the colab notebook) using :

```
import graphviz
data = tree.export_graphviz(regr)
graph = graphviz.Source(data, format="png")
graph
```