TW4: Decision Tree Classification

Submit a summary of your learning to Canvas. Your document should include:

- Full names of your team members who work on the assignment.
  - Xiaomei Xie
  - Lili Hao
- URL links to the notebook of each student on GitHub repo.
  - https://github.com/xiaomeiX/TW4-DT
  - https://github.com/lhaoSeattleu/TW4-DT
- A summary of what you learned from the teamwork assignment.

In supervised algorithms, we use the Gini Impurity or Entropy to split nodes since data labels are known. In TW4, we use "entropy" and "Gini" two method for our decision tree. Both Gini Impurity and Entropy are criteria to split a node in a decision tree.

Two metrics for choosing how to split a tree. Gini measurement is the probability of a random sample being classified incorrectly if we randomly pick a label according to the distribution in a branch.

Entropy is a measurement of information (or rather lack thereof). You calculate the information gain by making a split. Which is the difference in entripies. This measures how you reduce the uncertainty about the label.

change the following parameters and observe the changes.

Change max-tree-depth would slightly change the accuracy result. If a Decision Tree is overfitting the training set, is it a good idea to try decreasing maximum tree depth, which called regualiztion.

Part 2:

1. Look through an example list at: (Links to an external site.) https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset (Links to an external site.)

2. Discuss about different ways to handle the following types of data for decision tree classification. (Links to an external site.)

- text data (strings): in the case a dataset includes non-numerical data continuous data like age, weight, income, etc.

*** Notes: This will be also a reference for our next problem-solving assignment.

different ways to handle the following types of data for decision tree classification: -- Lili

In the article “Introduction to Decision Trees (Titanic dataset)”, the dataset dealt with the missing data, transformed text data into numerical values, and created new potential meaningful features based on existing features. So, further analysis, such as Pearson Correlation, heatmap, and decision tree can be construct based on the cleaned dataset.

The original dataset is as below:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

Impute the missing data:

The missing data first is filled with different reasonable values.

```
# Remove all NULLS in the Embarked column
for dataset in full_data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
# Remove all NULLS in the Fare column
for dataset in full_data:
    dataset['Fare'] = dataset['Fare'].fillna(train['Fare'].median())

# Remove all NULLS in the Age column
for dataset in full_data:
    age_avg = dataset['Age'].mean()
    age_std = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()
    age_null_random_list = np.random.randint(age_avg - age_std, age_avg + age_std, size=age_nul
l_count)
    # Next line has been improved to avoid warning
    dataset.loc[np.isnan(dataset['Age']), 'Age'] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)
```

Transformation:

Then, the text data is transformed into numeric values to represent different class:

- A new feature, “title” is created based on name.

```
for dataset in full_data:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr',
'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')

    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
```

- A new feature, “FamilySize”, is created to combine SibSp and Parch.

```
# Create new feature FamilySize as a combination of SibSp and Parch
for dataset in full_data:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
```

A new feature, “isAlone” is further created based on Familysize

```
# Create new feature IsAlone from FamilySize
for dataset in full_data:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
```

- A new feature, “Has_Cabin” is created as a Boolean value from the text data.

```
# Feature that tells whether a passenger had a cabin on the Titanic
train['Has_Cabin'] = train["Cabin"].apply(lambda x: 0 if type(x) == float else 1)
test['Has_Cabin'] = test["Cabin"].apply(lambda x: 0 if type(x) == float else 1)
```

- Sex, title, embarked, fare, and age are further categorized into classes.

```
for dataset in full_data:
    # Mapping Sex
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

    # Mapping titles
    title_mapping = {"Mr": 1, "Master": 2, "Mrs": 3, "Miss": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int)

    # Mapping Fare
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare']
= 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   = 2
    dataset.loc[ dataset['Fare'] > 31, 'Fare']
= 3
    dataset['Fare'] = dataset['Fare'].astype(int)

    # Mapping Age
    dataset.loc[ dataset['Age'] <= 16, 'Age']                                          = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age'] ;
```

Then, unnecessary columns are dropped.

```
# Feature selection: remove variables no longer containing relevant information
drop_elements = ['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp']
train = train.drop(drop_elements, axis = 1)
test  = test.drop(drop_elements, axis = 1)
```

- PClass: this stays the same.
- Survived: This class is in binary format. It indicates whether the passenger is survived or not. No additional formatting is necessary for this field.

	Survived	Pclass	Sex	Age	Parch	Fare	Embarked	Has_Cabin	FamilySize	IsAlone	Title
0	0	3	1	1	0	0	0	0	2	0	1
1	1	1	0	2	0	3	1	1	2	0	3
2	1	3	0	1	0	1	0	0	1	1	4

After all this, the dataset is much cleaner than before, with only numerical values and potential meaningful features. So, on top of those values, we can do further analysis, such as Pearson Correlation and heatmap.

Decision Tree:

For this dataset, Title and Sex are both attribute in the dataset. But Title is therefore more useful than *Sex.* Because *Title* implicitly includes information about *Sex* in most cases. T*he author uses Gini decision tree* learning algorithms to find the best split for each node of the tree. As result, Sex gets negative result. *Title* feature is slightly better at reducing the Gini Impurity than *Sex*. The most "Informative" attribute is the one maximized Gain.

For the decision trees, the 'max_depth' parameter determines the maximum number of attributes the model is going to use for each prediction. When the max_depth value is changed the Average Accuracy will slightly different. We can use max_depth to avoid the over-fitting our model, which called regulization.

Submission(s)

Each student should make individual submissions.

- **Part 1**:
    - Push an updated notebook file to his/her/their Git repo.
        - **You do not need to submit any notebook files to Canvas**.
        - I will visit your Github to check the file.
- **Part 2**:
    - Submit a summary of your learning to Canvas. Your document should include:
        - Full names of your team members who work on the assignment.
-
    -
        - URL links to the notebook of each student on GitHub repo.
        - A summary of what you learned from the teamwork assignment.