

# IEEE WIE Recruitments: Technical and Projects

## Machine Learning Task (Report)

Name: Rohini Das

Reg.No.: 20BCB0098

Question Number: 2

Date: 06.02.2022

---

AIM: Using an appropriate machine learning algorithm, to make a model which takes the person's input as its age, gender and fare and predicts whether the person survived or died when the RMS Titanic sank in the North Atlantic ocean.

ALGORITHM: Decision Tree Algorithm has been used to build an ML model to accomplish the given task. I used the CART Decision Tree Algorithm. It uses the “gini” as its criterion instead of entropy.

CART (Classification and Regression Trees) can be used for both classification and regression problems. With classification, we attempt to predict a class label, e.g., whether the passengers survived or not.

- In the CART decision tree, the nodes are split into sub-nodes on the basis of a threshold value of Gini Index.
- The root node is taken as the training set and is split into two by considering the best attribute and threshold value.
- Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.

DATASET USED: Titanic Dataset from Kaggle ( <https://www.kaggle.com/c/titanic-dataset/data> )

- The training dataset is used to build our machine learning models. For the training set, the outcome is given for each passenger i.e. “survived”. Our model is based on “features” like passengers’ age, gender, and fare.
- The test dataset is used to see how well our model performs on unseen data. For the test dataset, the final survival status for each passenger is not given. Our model would predict these outcomes for the unseen data.

PROGRAM & OUTPUT: I have written well-documented code, explaining each and every code block and also the general flow of execution of my solution for the task. For reference I have attached a copy of the Colab notebook consisting of the documented code and explanations, below.

## Import Modules

```
# Import necessary modules

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
```

## Reading Dataset

```
# Import the titanic train dataset as a Pandas dataframe

df = pd.read_csv("titanic_train.csv")
df.head()
```

```
passenger_id  pclass    name    sex  age  sibsp  parch  ticket    fare  cabin  embarked  boat  bc
```

```
# Since the model takes the input as the person's age, gender and fare: discard the rest of the attributes
```

```
df.drop(['passenger_id', 'pclass', 'name', 'sibsp', 'parch', 'ticket', 'cabin', 'embarked', 'boat', 'body', 'home.dest'], axis='columns', inplace=True)
```

	sex	age	fare	survived	
0	female	NaN	7.7333	1	
1	male	38.0	8.6625	0	
2	female	30.0	24.1500	0	
3	female	54.0	23.0000	1	
4	male	40.0	13.0000	0	
...	...	...	...	...	
845	male	55.0	50.0000	0	
846	male	58.0	29.7000	0	
847	female	24.0	26.0000	1	
848	female	3.0	13.7750	0	
849	male	52.0	13.0000	0	

850 rows × 4 columns

## Data Preprocessing for Training Dataset

Since the training dataset has string values and empty values, we need to format it before using it to build our model

```
# For the input dataframe, convert the string values to integer.
```

```
# (could also be done using LabelEncoder)
```

```
inputs = df.drop('survived', axis='columns')
inputs.sex = inputs.sex.map({'male': 1, 'female': 2})
inputs
```

	sex	age	fare
0	2	NaN	7.7333
1	1	38.0	8.6625
2	2	30.0	24.1500
3	2	54.0	23.0000
4	1	40.0	13.0000
...	...	...	...
845	1	55.0	50.0000
846	1	58.0	29.7000
847	2	24.0	26.0000
848	2	3.0	13.7750
849	1	52.0	13.0000

850 rows × 3 columns

```
# Check whether there are null values present
```

```
inputs.isna().sum()
```

```
sex      0
age     174
fare      1
dtype: int64
```

```
# Since there are null values are present for age and fare,
# fill the empty values with the mean of the existing data.
```

```
inputs.age = inputs.age.fillna(inputs.age.mean())
inputs.fare = inputs.fare.fillna(inputs.fare.mean())
inputs
```

	sex	age	fare
0	2	29.519847	7.7333
1	1	38.000000	8.6625
2	2	30.000000	24.1500
3	2	54.000000	23.0000
4	1	40.000000	13.0000
...	...	...	...
845	1	55.000000	50.0000
846	1	58.000000	29.7000
847	2	24.000000	26.0000
848	2	3.000000	13.7750
849	1	52.000000	13.0000

850 rows × 3 columns

```
# Create the target dataframe
```

```
target = df.survived
target
```

0	1
1	0

```
2      0
3      1
4      0
..
845    0
846    0
847    1
848    0
849    0
```

Name: survived, Length: 850, dtype: int64

## Building the Model

```
# Split the dataset into training dataset and test data for building the model
```

```
X_train, X_test, y_train, y_test = train_test_split(inputs.values,target,test_size=0.2)
```

```
#Create the model using the Decision Tree Classifier
```

```
model = tree.DecisionTreeClassifier()
```

```
# To display the settings of the classifier function
```

```
from sklearn import set_config
set_config(print_changed_only=False)
```

```
# Fit the dataset onto the model
```

```
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        random_state=None, splitter='best')
```

## Prediction and Accuracy

```
# Predict a sample data entry  
  
model.predict([[2,30,24.15]])  
  
array([0])
```

```
# Display the score of the model  
  
model.score(X_test, y_test)  
  
0.8117647058823529
```

## Testing of given Test Data

Since the test dataset has string values and empty values, we need to format it before using it to test our model

```
# Import the test data as a Pandas dataframe  
  
df_test = pd.read_csv("titanic_test.csv")  
df_test
```

	passenger_id	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	body	home.dest
0	295	1	Thayer, Mr. John Borland Jr	male	17.0	0	2	17421	110.8833	C70	C	B	NaN	Haverford, PA
1	1150	3	Risien, Mr. Samuel Beard	male	NaN	0	0	364498	14.5000	NaN	S	NaN	NaN	NaN
2	89	1	Davidson, Mr. Thornton	male	31.0	1	0	F.C. 12750	52.0000	B71	S	NaN	NaN	Montreal, PQ
3	1063	3	Nirva, Mr. Iisakki Antino Aijo	male	41.0	0	0	SOTON/O2 3101272	7.1250	NaN	S	NaN	NaN	Finland Sudbury, ON
4	1020	3	Minkoff, Mr. Lazar	male	21.0	0	0	349211	7.8958	NaN	S	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
454	1194	3	Sdycoff, Mr. Todor	male	NaN	0	0	349222	7.8958	NaN	S	NaN	NaN	NaN

```
# Create a separate dataframe for the given Passenger IDs
```

```
p_id = df_test['passenger_id']
p_id
```

```
0      295
1     1150
2       89
3     1063
4     1020
...
454    1194
455     403
456     108
457     510
```




458      1265

Name: passenger\_id, Length: 459, dtype: int64

```
# Drop the unnecessary attributes from the Test Dataframe; keep age, gender, fare
```

```
df_test.drop(['passenger_id', 'pclass', 'name', 'sibsp', 'parch', 'ticket', 'cabin', 'embarked', 'boat', 'body', 'home.dest'], axis='columns', inplace=True)
df_test
```

	sex	age	fare	
0	male	17.0	110.8833	
1	male	NaN	14.5000	
2	male	31.0	52.0000	
3	male	41.0	7.1250	
4	male	21.0	7.8958	
...	...	...	...	
454	male	NaN	7.8958	
455	male	23.0	13.0000	
456	female	NaN	110.8833	
457	male	16.0	10.5000	
458	female	10.0	24.1500	

459 rows × 3 columns

## Data Preprocessing for Test Dataset

```
# Convert the string values to integers
```

```
df_test.sex = df_test.sex.map({'male': 1, 'female': 2})
df_test
```

	sex	age	fare
0	1	17.0	110.8833
1	1	NaN	14.5000
2	1	31.0	52.0000
3	1	41.0	7.1250
4	1	21.0	7.8958
...	...	...	...
454	1	NaN	7.8958
455	1	23.0	13.0000
456	2	NaN	110.8833
457	1	16.0	10.5000
458	2	10.0	24.1500



459 rows × 3 columns

```
# Check for null values present
```

```
df_test.isna().sum()
```

```
sex      0
age      89
fare      0
dtype: int64
```

```
# Since there are null values are present for age,
# fill the empty values with the mean of the existing data.
```



```

0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0,
0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0,
0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1])

```

```
# Store the predicted result as a dataframe
```

```
df_result = pd.DataFrame(pred, columns = ['survived_status'])
df_result
```

	survived_status
0	0
1	1
2	0



```
# Create a separate column to store the final
# predicted result against the respective Passenger IDs
```

```
df_result['passenger_id'] = p_id
df_result
```

	survived_status	passenger_id
0	0	295
1	1	1150
2	0	89
3	0	1063
4	0	1020
...	...	...
454	1	1194
455	0	403
456	1	108
457	0	510
458	1	1265



459 rows × 2 columns

---

✓ 0s completed at 12:00 PM

