ACTIVITY NO. 4 | Advanced Data Analytics and Machine Learning **Name** | Gallano,Genaro Jr. U. **Section** | CPE32S1 **Date Performed**: | 6/20/2024 **Date Submitted**: | 6/20/2024 **Instructor**: | Engr. Roman,Richard

#PART 1 Part 1: Import the Libraries and Data

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
T_test = pd.read_csv('/content/titanic_test.csv')
T_train = pd.read_csv('/content/titanic_train.csv')
```

Part 2 - Plot the Data

```
T_test.head(10)
```


```
{"summary":"{\n  \"name\": \"T_test\",\n  \"rows\": 418,\n  \"fields\": [\n
{\n      \"column\": \"PassengerId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 120,\n        \"min\": 892,\n
\"max\": 1309,\n        \"num_unique_values\": 418,\n        \"samples\": [\n
1213,\n          1216,\n          1280\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n
{\n      \"column\": \"Pclass\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
2,\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Name\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 418,\n        \"samples\": [\n          \"Krekorian,
Mr. Neshan\",\n          \"Kreuchen, Miss. Emilie\",\n          \"Canavan,
Mr. Patrick\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Sex\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n          \"female\",\n
\"male\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Age\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
14.181209235624422,\n        \"min\": 0.17,\n        \"max\": 76.0,\n
\"num_unique_values\": 79,\n        \"samples\": [\n          10.0,\n
34.5\n        ],\n        \"semantic_type\": \"\",\n        \"description\":
\"\"\n      }\n    },\n    {\n      \"column\": \"SibSp\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 8,\n        \"num_unique_values\": 7,\n
\"samples\": [\n          0,\n          1\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n
{\n      \"column\": \"Parch\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 0,\n        \"max\": 9,\n
```

\"num_unique_values\": 8,\n   \"samples\": [\n   1,\n
6\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   },\n   {\n   \"column\": \"Ticket\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 363,\n   \"samples\": [\n   \"2673\",\n
\"W./C. 6607\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
55.907576179973844,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 169,\n   \"samples\": [\n   41.5792,\n
57.75\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   },\n   {\n   \"column\": \"Cabin\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 76,\n   \"samples\": [\n   \"A21\",\n
\"E45\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Embarked\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"Q\",\n
\"S\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   }\n]\n}","variable_name":"T_test","type":"dataframe"}

```
T_test.head(10)
```

{"summary":"{\n \"name\": \"T_test\",\n \"rows\": 418,\n \"fields\": [\n
{\n   \"column\": \"PassengerId\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 120,\n   \"min\": 892,\n
\"max\": 1309,\n   \"num_unique_values\": 418,\n   \"samples\": [\n
1213,\n   1216,\n   1280\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   3,\n
2,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 418,\n   \"samples\": [\n   \"Krekorian,
Mr. Neshan\",\n   \"Kreuchen, Miss. Emilie\",\n   \"Canavan,
Mr. Patrick\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   \"female\",\n
\"male\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
14.181209235624422,\n   \"min\": 0.17,\n   \"max\": 76.0,\n
\"num_unique_values\": 79,\n   \"samples\": [\n   10.0,\n
34.5\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n

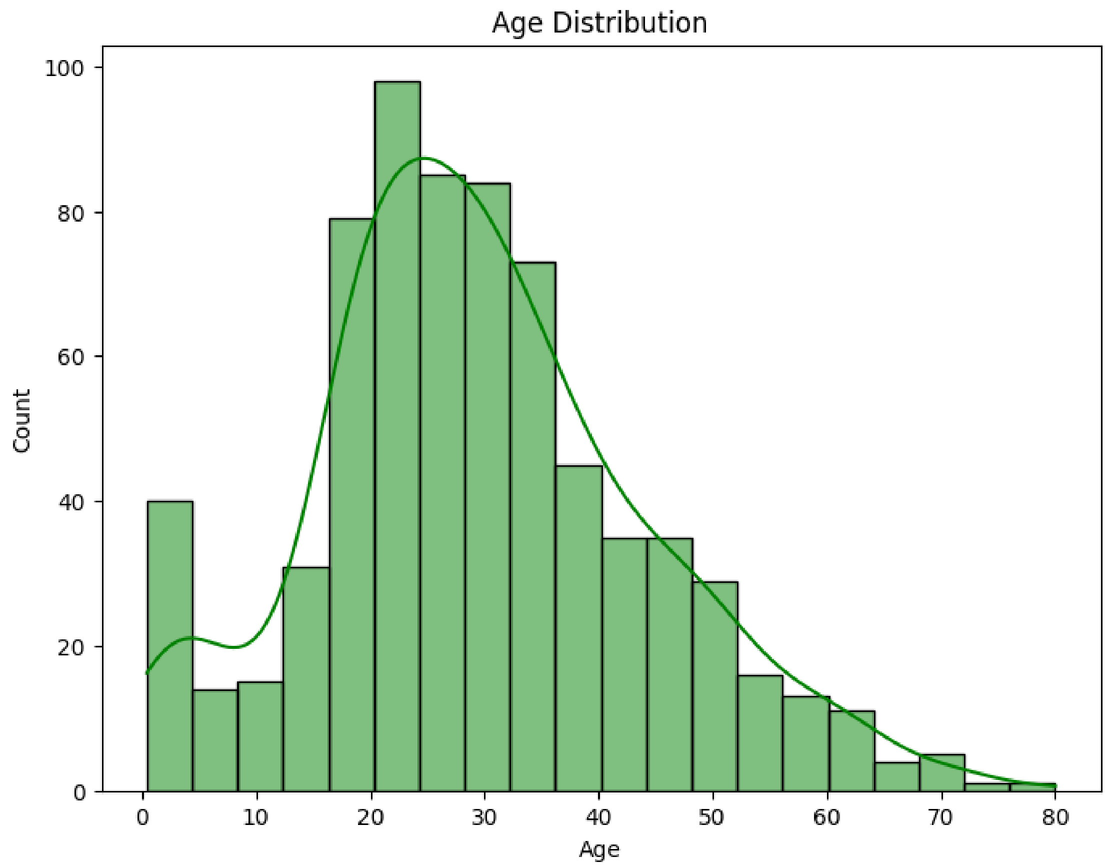
\"samples\": [\n 0,\n 1\n],\n
\"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n
{\n \"column\": \"Parch\",\n \"properties\": {\n \"dtype\":
\"number\",\n \"std\": 0,\n \"min\": 0,\n \"max\": 9,\n
\"num_unique_values\": 8,\n \"samples\": [\n 1,\n
6\n],\n \"semantic_type\": \"\",\n \"description\":
\"\"\n }\n },\n {\n \"column\": \"Ticket\",\n
\"properties\": {\n \"dtype\": \"string\",\n
\"num_unique_values\": 363,\n \"samples\": [\n \"2673\",\n
\"W./C. 6607\"\n],\n \"semantic_type\": \"\",\n
\"description\": \"\"\n }\n },\n {\n \"column\": \"Fare\",\n
\"properties\": {\n \"dtype\": \"number\",\n \"std\":
55.907576179973844,\n \"min\": 0.0,\n \"max\": 512.3292,\n
\"num_unique_values\": 169,\n \"samples\": [\n 41.5792,\n
57.75\n],\n \"semantic_type\": \"\",\n \"description\":
\"\"\n }\n },\n {\n \"column\": \"Cabin\",\n
\"properties\": {\n \"dtype\": \"category\",\n
\"num_unique_values\": 76,\n \"samples\": [\n \"A21\",\n
\"E45\"\n],\n \"semantic_type\": \"\",\n
\"description\": \"\"\n }\n },\n {\n \"column\":
\"Embarked\",\n \"properties\": {\n \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n \"samples\": [\n \"Q\",\n
\"S\"\n],\n \"semantic_type\": \"\",\n \"description\":
\"\"\n }\n }\n]\n}","variable_name":"T_test","type":"dataframe"}

```
T_test.describe()
```

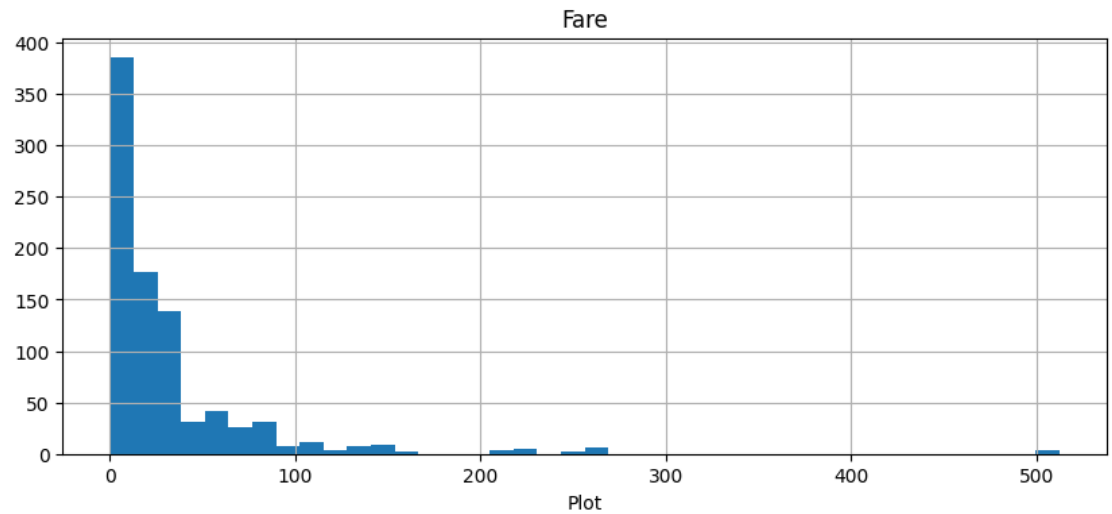
{"summary":"{\n \"name\": \"T_test\",\n \"rows\": 8,\n \"fields\": [\n
{\n \"column\": \"PassengerId\",\n \"properties\": {\n
\"dtype\": \"number\",\n \"std\": 412.1232851470217,\n \"min\":
120.81045760473994,\n \"max\": 1309.0,\n \"num_unique_values\":
7,\n \"samples\": [\n 418.0,\n 1100.5,\n
1204.75\n],\n \"semantic_type\": \"\",\n
\"description\": \"\"\n }\n },\n {\n \"column\":
\"Pclass\",\n \"properties\": {\n \"dtype\": \"number\",\n
\"std\": 147.0758997861715,\n \"min\": 0.8418375519640503,\n
\"max\": 418.0,\n \"num_unique_values\": 5,\n \"samples\": [\n
2.2655502392344498,\n 3.0,\n 0.8418375519640503\n
],\n \"semantic_type\": \"\",\n \"description\": \"\"\n
}\n },\n {\n \"column\": \"Age\",\n \"properties\": {\n
\"dtype\": \"number\",\n \"std\": 109.15868834351015,\n
\"min\": 0.17,\n \"max\": 332.0,\n \"num_unique_values\": 8,\n
\"samples\": [\n 30.272590361445783,\n 27.0,\n
332.0\n],\n \"semantic_type\": \"\",\n \"description\":
\"\"\n }\n },\n {\n \"column\": \"SibSp\",\n
\"properties\": {\n \"dtype\": \"number\",\n \"std\":
147.28745840271156,\n \"min\": 0.0,\n \"max\": 418.0,\n
\"num_unique_values\": 6,\n \"samples\": [\n 418.0,\n
0.4473684210526316,\n 8.0\n],\n \"semantic_type\":
\"\",\n \"description\": \"\"\n }\n },\n {\n

\"column\": \"Parch\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 147.29373273558997,\n   \"min\": 0.0,\n
\"max\": 418.0,\n   \"num_unique_values\": 5,\n   \"samples\": [\n
0.3923444976076555,\n   9.0,\n   0.9814288785371691\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 206.20808392426468,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\":
8,\n   \"samples\": [\n   35.627188489208635,\n
14.4542,\n   417.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
plt.figure(figsize=(8, 6))
sns.histplot(T_train['Age'].dropna(), bins=20, kde=True, color='green')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.show()
```




```
T_train['Fare'].hist(bins=40,figsize=(10,4))
plt.title('Fare')
plt.xlabel('Plot')
plt.show()
```



Part 3: Perform Simple Linear Regression on the SURVIVAL feature column (you can check the internet on how you can perform simple linear regression)

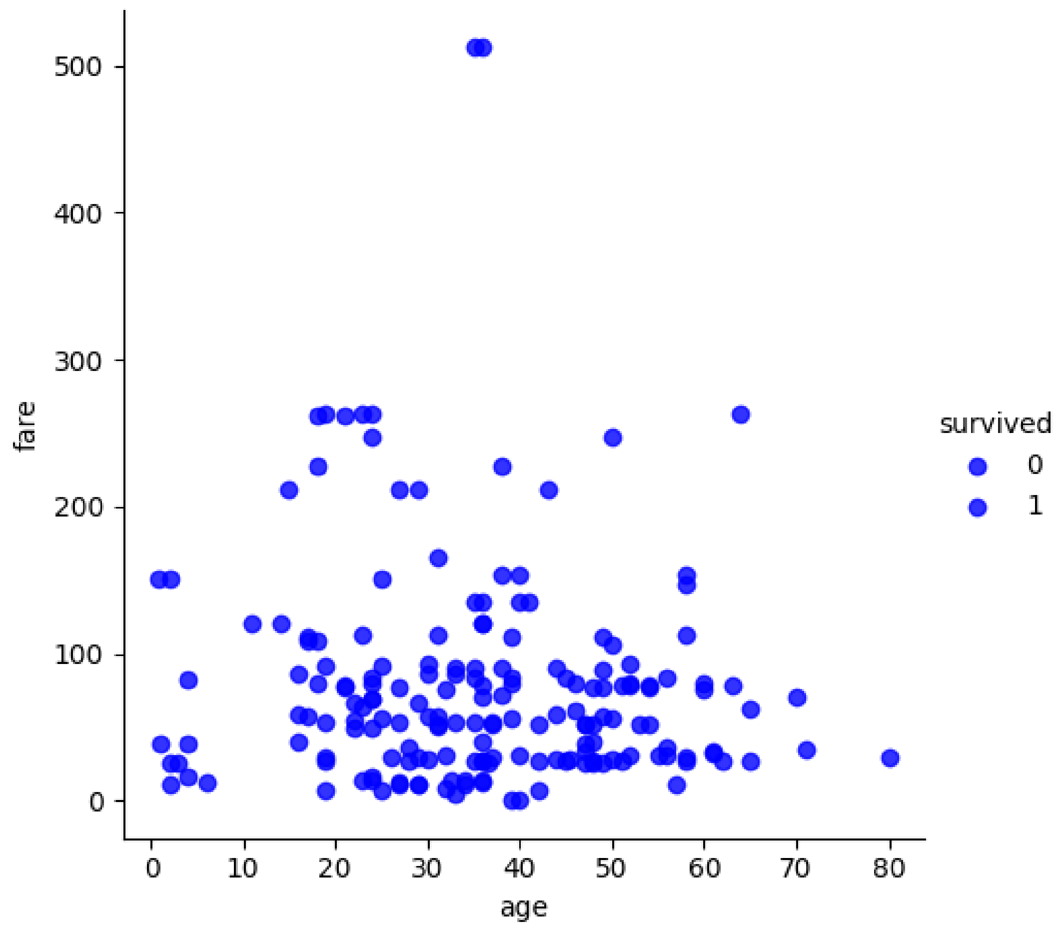
```
import seaborn as sns
import matplotlib.pyplot as plt

Titanic = sns.load_dataset('titanic')
Titanic = Titanic.dropna()
X = Titanic[['age', 'fare']]
y = Titanic['survived']

model = LinearRegression()
plt.figure(figsize=(8, 6))

sns.lmplot(x='age', y='fare', hue='survived', data=Titanic, fit_reg=False,
scatter_kws={'color': 'blue'})
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



Lab - Decision Tree Classification

Objectives

In this lab, you will use a decision tree classifier model to determine who survived the Titanic cruise ship disaster. **Part 1: Create a Decision Tree Classifier Part 2: Apply the Decison Tree Model Part 3: Evaluate the Decison Tree Model**

Scenario / Background

In this lab you will create a decision tree classifier that will work with a data set which contains the details about the more than 1300 hundred passengers who were onboard the passenger liner Titanic on its infamous maiden voyage.

Required Resources

- 1 PC with Internet access
- Python libraries: `pandas`, `sklearn`, and `IPython.display`
- Additonal application: Graphviz
- Datafiles: titanic-train.csv, titanic-test.csv, titanic_all.csv

Part 1: Create a Decision Tree Classifier

In this part of the lab, you will create a decision tree classifier that will learn from a labelled dataset.

The dataset contains the names and demographic details for each passenger. In addition, details of the passengers' trip are included. From this data, we can be build a decision tree that illustrates the factors that contributed to survivability, or lack of it, for the voyage.

The datasets contain the following variables:

With the data above, what kinds of questions can we ask about the factors that contributed to passengers surviving or perishing in the Titanic disaster?

Type your answers here

Step 1: Create the dataframe

a) Import pandas and the csv file

First, import pandas and create a dataframe from the Titanic training data set, which is stored in the titanic-train.csv file. Use the `pd.read_csv() method`.

```
#Code cell 1
#import pandas
import pandas as pd

#create a pandas dataframe called "training" from the titanic-train.csv file
T_train = pd.read_csv("/content/titanic_train.csv")
```

b) Verify the import and take a look at the data.

```
#Code cell 2
#verify the contents of the training dataframe using the pandas info()
method.
#training.head()
#training.dtypes
T_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
```

```
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Are there missing values in the data set?

Yes, there is missing.

```
#Code cell 3
#view the first few rows of the data
T_train.head()
```

{"summary":"{\n \"name\": \"T_train\",\n \"rows\": 891,\n \"fields\": [\n   {\n   \"column\": \"PassengerId\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 257,\n   \"min\": 1,\n   \"max\": 891,\n   \"num_unique_values\": 891,\n   \"samples\": [\n   710,\n   440,\n   841\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Survived\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   3,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 891,\n   \"samples\": [\n   \"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n   \"Kvillner, Mr. Johan Henrik Johannesson\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"female\",\n   \"male\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 14.526497332334042,\n   \"min\": 0.42,\n   \"max\": 80.0,\n   \"num_unique_values\": 88,\n   \"samples\": [\n   0.75,\n   22.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   0,\n

1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 681,\n   \"samples\": [\n   \"11774\",\n   \"248740\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 49.6934285971809,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\": 248,\n   \"samples\": [\n   11.2417,\n   51.8625\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 147,\n   \"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","variable_name":"T_train","type":"dataframe"}

Step 2: Prepare the Data for the Decision Tree Model.

a) Replace string data with numeric labels

We will use scikit-learn to create the decision trees. The decision tree model we will be using can only handle numeric data. The values for the Gender variable must be transformed into numeric representations. 0 will be used to represent "male" and 1 will represent "female."

In this code, a lambda expression is used with the `apply()` dataframe method. This lambda expression represents a function that uses a conditional statement to replace the text values in the columns with the appropriate numeric value. The lambda statement can be interpreted as "if the parameter toLabel equals 'male', return 0, if the value is something else, return 1." The `apply()` method will execute this function on the values in every row of the "Gender" column of the dataframe.

```
#code cell 4
T_train["Gender"] = T_train["Sex"].apply(lambda toLabel: 0 if toLabel ==
'male' else 1)
```

b) Verify that the Gender variable has been changed.

The output should show values of 0 or 1 for the Gender variable in the dataset.

```
#code cell 5
#view the first few rows of the data again
T_train.head()
```

{"summary":"{\n \"name\": \"T_train\",\n \"rows\": 891,\n \"fields\": [\n   {\n   \"column\": \"PassengerId\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 257,\n   \"min\": 1,\n

\"max\": 891,\n   \"num_unique_values\": 891,\n   \"samples\": [\n
710,\n   440,\n   841\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Survived\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n
\"num_unique_values\": 2,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   },\n   {\n   \"column\": \"Pclass\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n
\"samples\": [\n   3,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\":
\"string\",\n   \"num_unique_values\": 891,\n   \"samples\": [\n
\"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n
\"Kvillner, Mr. Johan Henrik Johannesson\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Sex\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n
\"female\",\n   \"male\"\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Age\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 14.526497332334042,\n   \"min\": 0.42,\n
\"max\": 80.0,\n   \"num_unique_values\": 88,\n   \"samples\": [\n
0.75,\n   22.0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n
\"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n
\"samples\": [\n   1,\n   0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Parch\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n   \"description\":
\"\"\n   }\n   },\n   {\n   \"column\": \"Ticket\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 681,\n   \"samples\": [\n   \"11774\",\n
\"248740\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
49.6934285971809,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n   11.2417,\n
51.8625\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Cabin\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 147,\n   \"samples\": [\n   \"D45\",\n
\"B49\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Embarked\",\n   \"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n   \"samples\": [\n   \"S\",\n

```
\"C\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\":
\"\"\n      }\n    },\n    {\n      \"column\": \"Gender\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0,\n
\"min\": 0,\n        \"max\": 1,\n        \"num_unique_values\": 2,\n
\"samples\": [\n          1,\n          0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n
]\n}","variable_name":"T_train","type":"dataframe"}
```


c) Address Missing Values in the Dataset

The output of the info() method above indicated that about 180 observations are missing the age value. The age value is important to our analysis. We must address these missing values in some way. While not ideal, we can replace these missing age values with the mean of the ages for the entire dataset.

This is done by using the `fillna()` method on the "Age" column in the dataset. The `fillna()` method will change the original dataframe by using the `inplace = True` argument.

```
#code cell 6
T_train["Age"].fillna(T_train["Age"].mean(), inplace=True)
```

d) Verify that the values have been replaced.

```
#code cell 7
#verify that the missing values for the age variable have been eliminated.
T_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  Gender       891 non-null    int64
dtypes: float64(2), int64(6), object(5)
memory usage: 90.6+ KB
```

What is the value that was used to replace the missing ages?

```
T_train["Age"].fillna(T_train["Age"].mean(), inplace=True)
```

```
T_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
 12  Gender       891 non-null    int64
dtypes: float64(2), int64(6), object(5)
memory usage: 90.6+ KB
```

Step 3: Train and Score the Decision Tree Model.

a) Create an array object with the variable that will be the target for the model.

The purpose of the model is to classify passengers as survivors or victims. The dataset identifies survivors and victims. The model will learn which input variable values are most likely to belong to victims and survivors, and then use that information to classify passengers from a unique test data set.

```
#code cell 8
#create the array for the target values
y_target = T_train["Survived"].values
```

b) Create an array of the values that will be the input for the model.

Only some of the features of the data are useful for creating the classifier tree. We create a list of the columns from the data that we want the classifier to use as the input variables and then create an array using the column name from that variable. The variable `X_input` holds the values for all the features that the model will use to learn how to make the classifications. After the model is trained, we will use this variable to assign these labels to the test data set.

```
#code cell 9
columns = ["Fare", "Pclass", "Gender", "Age", "SibSp"]
#create the variable to hold the features that the classifier will use
X_input = T_train[list(columns)].values
```

c) Create the learned model.

Import the decision tree module from the sklearn machine learning library. Create the classifier object clf_train. Then, use the fit() method of the classifier object, with the X_input and y_target variables as parameters, to train the model.

```
#code cell 10
#import the tree module from the sklearn library
from sklearn import tree
#create clf_train as a decision tree classifier object
clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
#train the model using the fit() method of the decision tree object.
#Supply the method with the input variable X_input and the target variable
y_target
clf_train = clf_train.fit(X_input, y_target)
```

d) Evaluate the model

Use the score() method of the decision tree object to display the percentage accuracy of the assignments made by the classifier. It takes the input and target variables as arguments.

```
#code cell 11
clf_train.score(X_input,y_target)
```

```
0.8226711560044894
```

This score value indicates that classifications made by the model should be correct approximately 82% of the time.

Step 6: Visualize the Tree

a) Create the intermediate file output

Import the sklearn.externals.six StringIO module which is used to output the characteristics of the decision tree to a file. We will create a Graphviz dot file which will allow us to export the results of the classifier into a format that can be converted into a graphic.

```
#code cell 12
from six import StringIO
with open("/content/sample_data/titanic.dot", 'w') as f:
  f = tree.export_graphviz(clf_train, out_file=f, feature_names=columns)
```

b) Install Graphviz

To visualize the decision tree, Graphviz needs to be installed from a terminal. The installation requires that a prompt be answered, which can't be done from a notebook code cell. Use the apt-get install graphviz command from the terminal command line to install this software.

c) Convert the intermediate file to a graphic

The dot file that was created above can be converted to a .png file with the graphiz dot renderer. This is a shell command, so use `!` before it to run it from this noteblook. The new titanic.png graphic file should appear in the directory that contains this notebook.

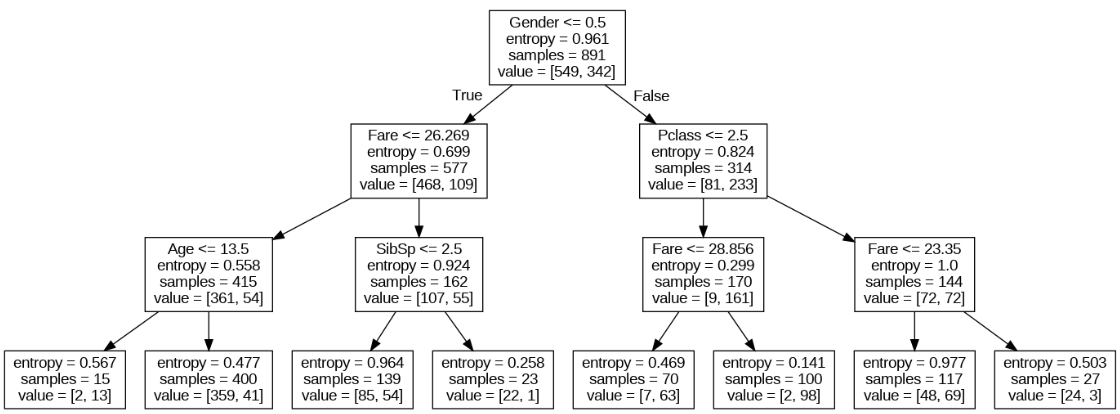
```
#code cell 13
#run the Graphviz dot command to convert the .dot file to .png
!dot -Tpng /content/sample_data/titanic.dot -o
/content/sample_data/titanic.png
```

d) Display the image

Now we will import the `Image` module from the `IPython.display` library. This will allow us to open and display an external graphics file on the notebook page. The `Image` function is used to display the file, with the .png file name as argument.

```
#code cell 14
#import the Image module from the Ipython.display libary
from IPython.display import Image

#display the decison tree graphic
Image("/content/sample_data/titanic.png")
```



e) Interpret the tree

From the tree, we can see several things. First, at the root of the tree is the Gender variable, indicating that it is the single most important factor in making the classification. The branches to the left are for Gender = 0 or male. Each root and intermediate node contains the decision factor, the entropy, and the number of passengers who fit the critierion at that point in the tree. For example, the root node indicates that there are 891 observations that make up the learning data set. At the next level, we can see that 577 people were male, and 314 were female. In the third level, at the far right, we can see that 415 people were male and paid a fare of less than 26.2686. Finally, the leaf nodes for that intermediate node indicate that 15 of these passengers were below the age of 13.5, and the other 400 were older than that age.

Finally, the elements in the value array indicate survival. The first value is the number of people who died, and the second is the number of survivors for each criterion. The root node tells us that out of our sample, 549 people died and 342 survived.

Entropy is a measure of noise in the decision. Noise can be viewed as uncertainty. For example, in nodes in which the decision results in equal values in the survival value array, the entropy is at its highest possible value, which is 1.0. This means that the model was unable to definitively make the classification decision based on the input variables. For values of very low entropy, the decision was much more clear cut, and the difference in the number of survivors and victims is much higher.

What describes the group that had the most deaths by number? Which group had the most survivors?

Based on the image, female group had the most survivor than male.

Part 2: Apply the Decision Tree Model

In this part of the lab, we will use the results of the learned decision tree model to label an unlabelled dataset of Titanic passengers. The decision tree will evaluate the features of each observation and label the observation as survived (label = 1) or died (label = 0).

Step 1: Import and Prepare the Data

In this step, you will import and prepare the data for analysis.

a) Import the data.

Name the dataframe "testing" and import the file titanic-test.csv.

```
#code cell 15
#import the file into the 'testing' dataframe.
T_test = pd.read_csv("/content/titanic_test.csv")
```

```
T_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  418 non-null    int64  
 1   Pclass       418 non-null    int64  
 2   Name         418 non-null    object 
 3   Sex          418 non-null    object 
 4   Age          332 non-null    float64
 5   SibSp        418 non-null    int64  
 6   Parch        418 non-null    int64  
 7   Ticket       418 non-null    object 
 8   Fare         417 non-null    float64
 9   Cabin        91 non-null     object 
```

```
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

How many records are in the data set?

- 418

Which important variables(s) are missing values and how many are missing?

- Fare 1 and Age 86

b) Use a lambda expression to replace the "male" and "female" values with 0 for male and 1 for female..

```
#code cell 16
#replace the Gender labels in the testing dataframe
# Hint: look at code cell 4
T_test["Sex"] = T_test["Sex"].apply(lambda toLabel: 0 if toLabel == 'male'
else 1)
```

c) Replace the missing age values with the mean of the ages.

```
#code cell 17
#Use the fillna method of the testing dataframe column "Age"
#to replace missing values with the mean of the age values.
T_test["Age"].fillna(T_test["Age"].mean(), inplace=True)

T_test.head()
```

{"summary":"{\n \"name\": \"T_test\",\n \"rows\": 418,\n \"fields\": [\n
{\n   \"column\": \"PassengerId\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 120,\n   \"min\": 892,\n
\"max\": 1309,\n   \"num_unique_values\": 418,\n   \"samples\": [\n
1213,\n   1216,\n   1280\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n
\"num_unique_values\": 3,\n   \"samples\": [\n   3,\n
2,\n   1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 418,\n   \"samples\": [\n   \"Krekorian,
Mr. Neshan\",\n   \"Kreuchen, Miss. Emilie\",\n   \"Canavan,
Mr. Patrick\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n
\"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n
\"samples\": [\n   1,\n   0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n
{\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 12.634534168325064,\n   \"min\": 0.17,\n
\"max\": 76.0,\n   \"num_unique_values\": 80,\n   \"samples\": [\n

28.0,\n 34.5\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"SibSp\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0,\n \"min\": 0,\n \"max\": 8,\n \"num_unique_values\": 7,\n \"samples\": [\n 0,\n 1\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Parch\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0,\n \"min\": 0,\n \"max\": 9,\n \"num_unique_values\": 8,\n \"samples\": [\n 1,\n 6\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Ticket\",\n \"properties\": {\n \"dtype\": \"string\",\n \"num_unique_values\": 363,\n \"samples\": [\n \"2673\",\n \"W./C. 6607\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Fare\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 55.840500479541035,\n \"min\": 0.0,\n \"max\": 512.3292,\n \"num_unique_values\": 170,\n \"samples\": [\n 41.5792,\n 57.75\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Cabin\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 76,\n \"samples\": [\n \"A21\",\n \"E45\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"Embarked\",\n \"properties\": {\n \"dtype\": \"category\",\n \"num_unique_values\": 3,\n \"samples\": [\n \"Q\",\n \"S\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n }\n]\n}","variable_name":"T_test","type":"dataframe"}

```
T_test["Fare"].fillna(T_test["Fare"].mean(), inplace=True)
```

d) Verify that the values have been replaced.

Check that the missing values have been filled and that the Gender labels are 0 and 1.

```
T_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  418 non-null    int64
 1   Pclass       418 non-null    int64
 2   Name         418 non-null    object
 3   Sex          418 non-null    int64
 4   Age          418 non-null    float64
 5   SibSp        418 non-null    int64
 6   Parch        418 non-null    int64
 7   Ticket       418 non-null    object
 8   Fare         418 non-null    float64
```

```
 9   Cabin        91 non-null     object
 10  Embarked     418 non-null    object
dtypes: float64(2), int64(5), object(4)
memory usage: 36.0+ KB
```

```
T_test.head()
```

{"summary":"{\n \"name\": \"T_test\",\n \"rows\": 418,\n \"fields\": [\n   {\n   \"column\": \"PassengerId\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 120,\n   \"min\": 892,\n   \"max\": 1309,\n   \"num_unique_values\": 418,\n   \"samples\": [\n   1213,\n   1216,\n   1280\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   3,\n   2,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 418,\n   \"samples\": [\n   \"Krekorian, Mr. Neshan\",\n   \"Kreuchen, Miss. Emilie\",\n   \"Canavan, Mr. Patrick\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12.634534168325064,\n   \"min\": 0.17,\n   \"max\": 76.0,\n   \"num_unique_values\": 80,\n   \"samples\": [\n   28.0,\n   34.5\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 9,\n   \"num_unique_values\": 8,\n   \"samples\": [\n   1,\n   6\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 363,\n   \"samples\": [\n   \"2673\",\n   \"W./C. 6607\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 55.840500479541035,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\": 170,\n   \"samples\": [\n   41.5792,\n   57.75\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Cabin\",\n

\"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 76,\n   \"samples\": [\n   \"A21\",\n   \"E45\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 3,\n   \"samples\": [\n   \"Q\",\n   \"S\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","variable_name":"T_test","type":"dataframe"}

Step 2: Label the testing dataset

In this step, you will apply the learned model to the testing dataset.

a) Create the array of input variables from the testing data set.

```
#code cell 19
columns = ["Fare", "Pclass", "Sex", "Age", "SibSp"]
#create the variable X_input to hold the features that the classifier will use
X_input = T_test[list(columns)].values
```

b) Apply the model to the testing data set.

Use the `predict()` method of the `clf_train` object that was trained to label the observations in the testing data set with the most likely survival classification. Provide the array of input variables from the testing data set as the parameter for this method.

```
#code cell 20
#apply the model to the testing data and store the result in a pandas dataframe.
#Use X_input as the argurment for the predict() method of the clf_train classifier object

target_labels = clf_train.predict(X_input)

#convert the target array into a pandas dataframe using the pd.DataFrame() method and target as argument
target_labels = pd.DataFrame({'Est_Survival':target_labels, 'Name':T_test['Name']})

#display the first few rows of the data set
target_labels.head()
```

{"summary":"{\n \"name\": \"target_labels\",\n \"rows\": 418,\n \"fields\": [\n   {\n   \"column\": \"Est_Survival\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Name\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 418,\n   \"samples\": [\n   \"Krekorian, Mr. Neshan\",\n   \"Kreuchen, Miss. Emilie\"\n

],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n }\n   }\n]\n}","variable_name":"target_labels","type":"dataframe"}

c) Evaluate the accuracy of the estimated labels

The ground truth for the survival of each passenger can be found in another file called `all_data.csv`. To select only the passengers contained in the testing dataset, we merge the `target_labels` dataframe and the `all_data` dataframe on the field `Name`. We then compare the estimated label with the ground truth dataframe and compute the accuracy of the learned model.

Part 3: Evaluate the Decision Tree Model

The `sklearn` library includes a module that can be used to evaluate the accurracy of the decision tree model. The `train_test_split()` method will divide the observations in whole data set into two randomly selected arrays of observations that makeup the testing and training datasets. After fitting the model to the training data, the trained model can be scored and the prediction accuracy compared for both the training and test datasets. It is desirable for the two scores to be close, but the accuracy for the test dataset is normally lower that for the training data set.

```
#code cell 21
#import the numpy library as np
import numpy as np
# Load data for all passengers in the variable all_data
all_data = pd.read_csv("/content/titanic_train.csv")
# Merging using the field Name as key, selects only the rows of the two
datasets that refer to the same passenger
testing_results = pd.merge(target_labels, all_data[['Name','Survived']],
on=['Name'])

# Compute the accuracy as a ratio of matching observations to total
osbervations. Store this in in the variable acc.
acc = np.sum(testing_results['Est_Survival'] == testing_results['Survived'])
/ float(len(testing_results))
# Print the result
```

Step 1: Import the data

This time we will import the data from a `csv` file, but we will specify the columns that we want to have appear in the dataframe. We will do this by passing an array-like list of column names to the `read_csv()` method `usecols` parameter. Use the following columns: 'Survived', 'Fare', 'Pclass', 'Gender', 'Age', and 'SibSP'. Each should be in quotes and the list should be square brackets. Name this dataframe `all_data`.

How many records are in the data set?

- There are 1308 records on the given dataset

Which important variables(s) are missing values and how many are missing?

- Age (263)

Step 2: Prepare the data.

a) Remove the "male" and "female" strings and replace them with 0 and 1 respectively.

```
#code cell 23
#Label the gender variable with 0 and 1
all_data["Gender"] = all_data["Gender"].apply(lambda toLabel: 0 if toLabel ==
'male' else 1)
```

c) Replace the missing age values with the mean of the age of all members of the data set.

```
#code cell 24
#replace missing Age values with the mean age
all_data["Age"].fillna(all_data["Age"].mean(), inplace=True)

#display the first few rows of the data set

all_data.head()
```

{"summary":"{\n \"name\": \"all_data\",\n \"rows\": 1308,\n \"fields\": [\n   {\n   \"column\": \"Survived\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Pclass\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   1,\n   2\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 12.860241791360886,\n   \"min\": 0.1667,\n   \"max\": 80.0,\n   \"num_unique_values\": 98,\n   \"samples\": [\n   6.0,\n   64.0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"SibSp\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n   \"num_unique_values\": 7,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 51.75866823917411,\n   \"min\": 0.0,\n   \"max\": 512.3292,\n   \"num_unique_values\": 281,\n   \"samples\": [\n   5.0,\n   9.35\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","variable_name":"all_data","type":"dataframe"}

Step 2: Create the input and output variables for the training and testing data.

The `sklearn` library includes modules that help with model selection. We will import from `sklearn.model_selection` the `train_test_split()` method. This method will automatically split the entire dataset, returning in total four numpy arrays, two for the features (test and validation) and two for the labels (test and validation). One parameter of the method specifies the proportion of observations to use for testing and training. Another parameter specifies a seed value that will be used to randomize assignment of the observation to testing or training. This is used so that another user can replicate your work by receiving the same assignments of observations to datasets. The syntax of the method is:

40% of the data will be used for testing. The random seed is set to 0.

The method returns four values. These values are the input varibles for training and testing data and the target variables for the training and testing data in that order. ##### a) Designate the input variables and output variables and generate the arrays.

```
T_test = pd.read_csv("/content/titanic_all.csv")
```

```
#code cell 25
#Import train_test_split() from the sklearn.model_selection libary
from sklearn.model_selection import train_test_split

#create the input and target variables as uppercase X and lowercase y. Reuse
the columns variable.
X = all_data[list(columns)].values
y = all_data["Survived"].values

#generate the four testing and training data arrays with the
train_test_split() method
X_train,X_test,y_train,y_test=train_test_split(X, y, test_size=0.40,
random_state=0)
```

```
all_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1308 entries, 0 to 1307
Data columns (total 6 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  1308 non-null   int64
 1   Pclass    1308 non-null   int64
 2   Gender    1308 non-null   int64
 3   Age       1308 non-null   float64
 4   SibSp     1308 non-null   int64
 5   Fare      1308 non-null   float64
dtypes: float64(2), int64(4)
memory usage: 61.4 KB
```

b) Train the model and fit it to the testing data.

Now the model can be fit again. The model will be trained using only the training datat, as selected by the `train_test_split` function.

```
#code cell 26
#create the training decision tree object
clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)

#fit the training model using the input and target variables
clf_train = clf_train.fit(X_train, y_train)
```

c) Compare models by scoring each.

Use the score() method of each decision tree object to generate scores.

```
#code cell 27
#score the model on the two datasets and store the scores in variables.
Convert the scores to strings using str()
train_score = str(clf_train.score(X_train,y_train))
test_score = str(clf_train.score(X_test,y_test))

#output the values in a test string
print('Training score = '+ train_score+' Testing score = '+test_score)
```

```
Training score = 0.8201530612244898 Testing score = 0.8053435114503816
```

```
# This is formatted as code
```

We have now compared the scores for the trained model on both test and validation data. As expected, the test accuracy score is close, but lower than the score for the training data. This is because normally, the model tends to overfit the training data, therefore the test score is a better evaluation of how the model is able to generalize outside of the training data.

```
!apt-get install pandoc
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pandoc is already the newest version (2.9.2.1-3ubuntu2).
pandoc set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
```

```
!pip install pypandoc
```

```
Collecting pypandoc
  Downloading pypandoc-1.13-py3-none-any.whl (21 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.13
```

```
import pypandoc

# Convert the IPython Notebook to DOCX
input_file =
'HOA4_1_Advanced_Data_Analytics_and_Machine_Learning_Diocera.ipynb'
output_file = 'output.docx'
pypandoc.convert_file(input_file, 'docx', outputfile=output_file)

# Download the output DOCX file
from google.colab import files
files.download(output_file)
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>
```

Part 4 For Further Study (optional)

If you have the time and are interested, you could try the following and see how the decision tree is affected.

1. Remove observations with missing Age values.

Using a mean to replace missing age values may affect the accuracy of the model. One approach to this might be to remove all observations with missing age values. Although this will decrease the size of the training dataset, it could improve accuracy.

2. Remove the input variables.

Another issue with this type of analysis is the identification of which input variables, or features, are essential to the accuracy of the classifier. One way to do this is to try running the classifier with different sets of input variables by editing the list of variables that is used to fit the model.

Conclusion

- In conclusion, this hands-on activity about analyzing different dataset using specific methods like the decision tree classification where it helps assessing the result of survival percentage based on a few titanic dataset. I have also learned the different factors that affects the survival of an individual or the import informations such as gender, age, fare, and other variables for assessing the level of survival positivity. I have successfully fulfilled the given objectives timely as this gives an improvement upon the usage of machine learning.