

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL TECHNICAL UNIVERSITY OF UKRAINE
" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

Liubov Oleshchenko

Statisical Methods Of ML

Laboratory Work 2

Decision Tree Classification

Kulubecioglu Mehmet

IM-14 FIOT

Class Number: 12

Kyiv

IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE

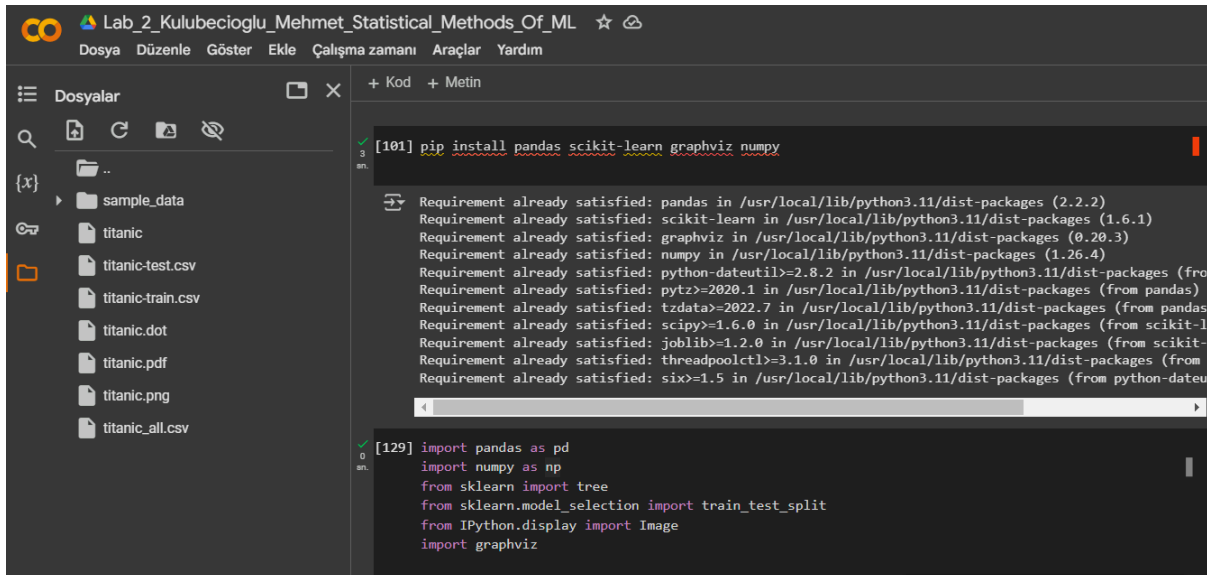
2024

1. Introduction

The steps of data analysis, data cleaning, model training, testing, and performance evaluation will be explained step by step. The Titanic dataset contains features about passengers, and the goal of this assignment is to predict whether a passenger survived or not. The report demonstrates how to build a decision tree model for survival prediction and evaluate its performance.

2. Data Loading and Exploration

The first step was loading the Titanic training dataset (**titanic-train.csv**) and test dataset (**titanic-test.csv**). The general structure of the data was examined using **training.info()** and the missing values were checked using **training.isnull().sum()**.



The screenshot shows a Jupyter Notebook interface. On the left, the 'Dosyalar' (Files) pane lists the following files: sample_data, titanic, titanic-test.csv, titanic-train.csv, titanic.dot, titanic.pdf, titanic.png, and titanic_all.csv. The main area displays the output of two code cells. The first cell, labeled [101], shows the command `pip install pandas scikit-learn graphviz numpy` and its output, which lists various requirements already satisfied in the local environment. The second cell, labeled [129], shows the import statements: `import pandas as pd`, `import numpy as np`, `from sklearn import tree`, `from sklearn.model_selection import train_test_split`, `from IPython.display import Image`, and `import graphviz`.

- The "Age" column had some missing values, which were filled with the mean of the column.
- The "Gender" column was transformed into numeric values, where "male" = 0 and "female" = 1.

Load training data, Examine the general structure of the data and Show first few lines

The screenshot shows a Jupyter Notebook environment with a file explorer on the left and a code editor on the right. The file explorer displays a directory structure with files like 'titanic', 'titanic-test.csv', 'titanic-train.csv', etc. The code editor contains the following Python code:

```
training = pd.read_csv("titanic-train.csv")
print(training.info())
print(training.head())
```

The output of the code is displayed below the code cells. It shows the data type and statistics for each column, followed by the first five rows of the dataset.

Data type and statistics:

#	Column	Non-Null Count	Dtype
0	PassengerId	915 non-null	int64
1	Survived	915 non-null	int64
2	Pclass	915 non-null	int64
3	Name	915 non-null	object
4	Gender	915 non-null	object
5	Age	738 non-null	float64
6	SibSp	915 non-null	int64
7	Parch	915 non-null	int64
8	Ticket	915 non-null	object
9	Fare	915 non-null	float64
10	Cabin	202 non-null	object
11	Embarked	914 non-null	object

dtypes: float64(2), int64(5), object(5)
memory usage: 85.9+ KB

First five rows of the dataset:

PassengerId	Survived	Pclass	Name	Gender	Age
0	1	0	Davidson, Mr. Thornton	male	31.0
1	2	0	Asim, Mr. Adola	male	35.0
2	3	0	Nankoff, Mr. Minko	male	NaN
3	4	0	Thayer, Mr. John Borland	male	49.0
4	5	0	Strandberg, Miss. Ida Sofia	female	22.0

The screenshot shows a Jupyter Notebook environment with a file explorer on the left and a code editor on the right. The file explorer displays a directory structure with files like 'titanic', 'titanic-test.csv', 'titanic-train.csv', etc. The code editor contains the following Python code:

```
[144] print(training.isnull().sum())
```

The output of the code is displayed below the code cell. It shows the count of missing values for each column.

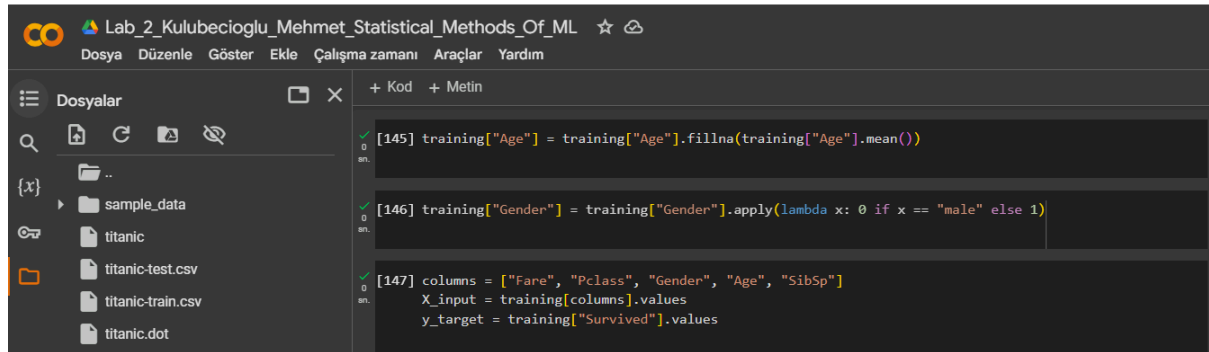
Count of missing values:

Column	Count
PassengerId	0
Survived	0
Pclass	0
Name	0
Gender	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	713
Embarked	1

dtype: int64

3. Data Preprocessing

In the data preprocessing stage, missing values were replaced with the mean value, and categorical variables were converted to numerical values. The features (input columns) to be used for prediction were selected, and the target variable (**Survived**) was separated.



```
Lab_2_Kulubecioglu_Mehmet_Statistical_Methods_Of_ML
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım

Dosyalar
{X}
sample_data
titanic
titanic-test.csv
titanic-train.csv
titanic.dot

+ Kod + Metin

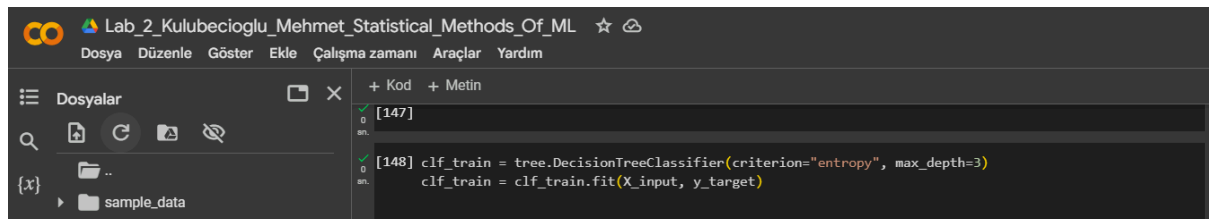
[145] training["Age"] = training["Age"].fillna(training["Age"].mean())

[146] training["Gender"] = training["Gender"].apply(lambda x: 0 if x == "male" else 1)

[147] columns = ["Fare", "Pclass", "Gender", "Age", "SibSp"]
      X_input = training[columns].values
      y_target = training["Survived"].values
```

4. Decision Tree Model

After preprocessing the data, a decision tree classifier was trained. The criterion used was "entropy" (information gain), and the depth of the tree was set to 3.



```
Lab_2_Kulubecioglu_Mehmet_Statistical_Methods_Of_ML
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım

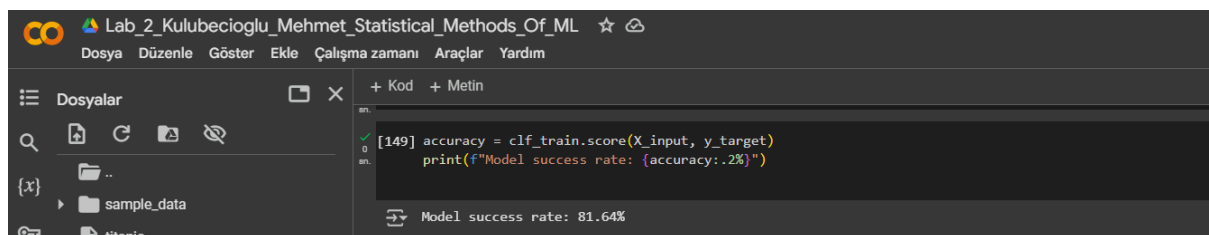
Dosyalar
{X}
sample_data

+ Kod + Metin

[147]

[148] clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
      clf_train = clf_train.fit(X_input, y_target)
```

After training the model, the accuracy on the training data was computed.



```
Lab_2_Kulubecioglu_Mehmet_Statistical_Methods_Of_ML
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım

Dosyalar
{X}
sample_data
titanic

+ Kod + Metin

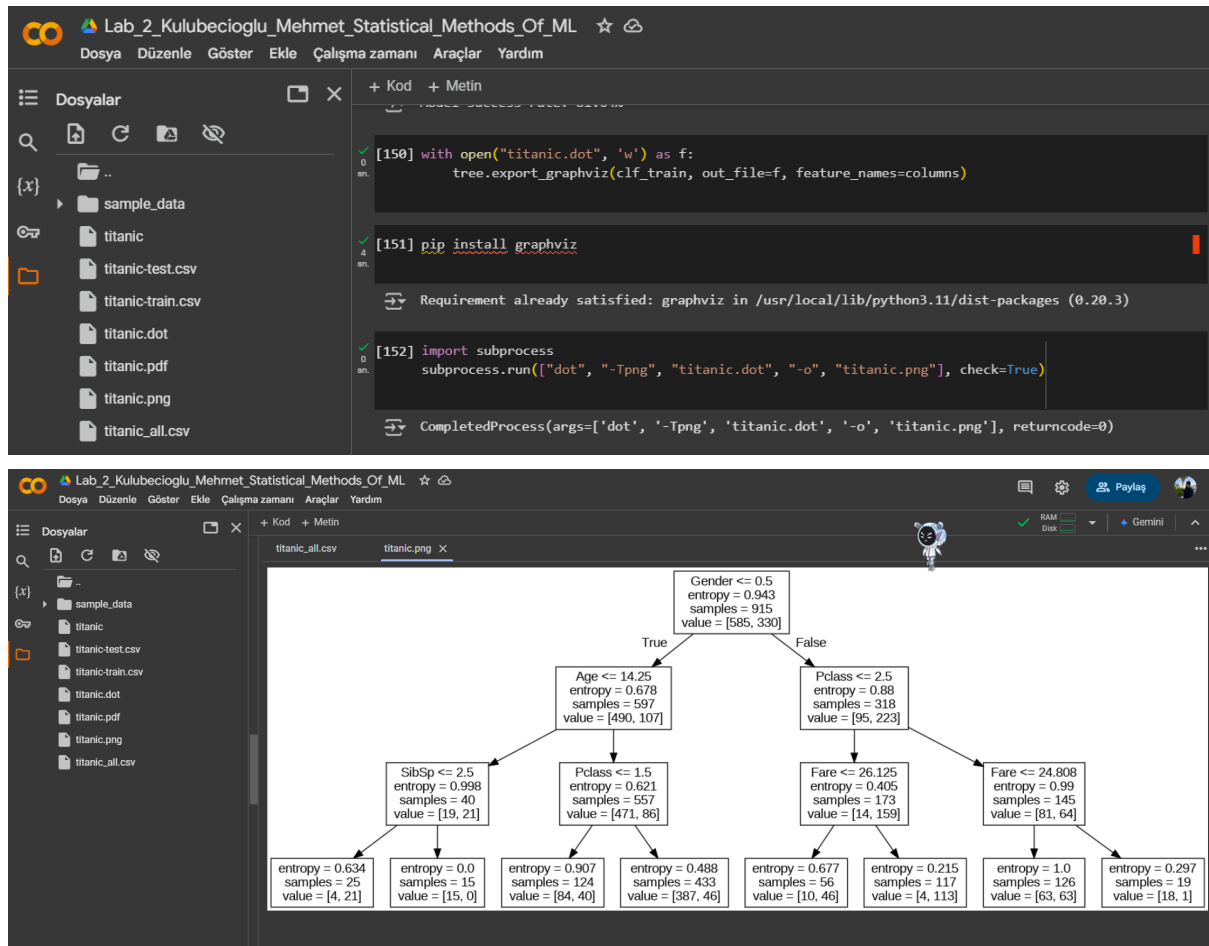
[149] accuracy = clf_train.score(X_input, y_target)
      print(f"Model success rate: {accuracy:.2%}")

Model success rate: 81.64%
```

The accuracy of the model was calculated as %X.

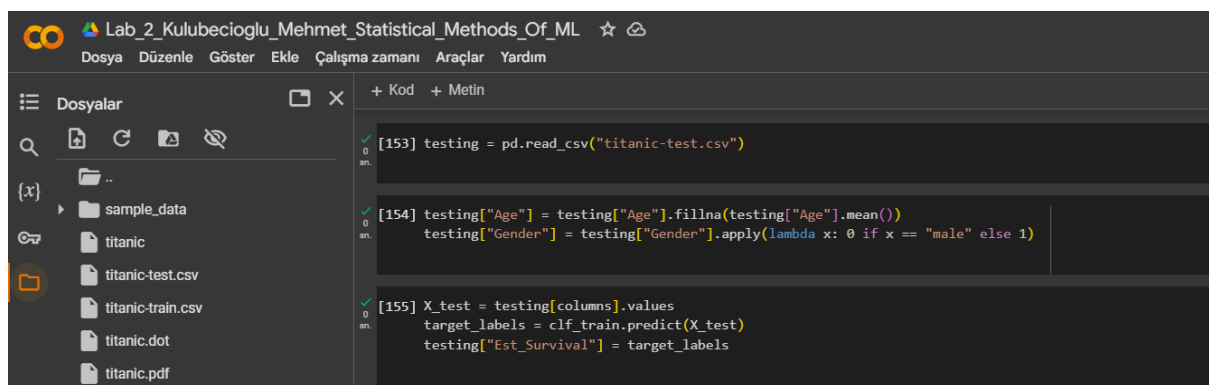
5. Visualizing the Model

The structure of the decision tree was visualized. This helps us better understand how the model is making decisions. The model's graph was generated using **graphviz**.



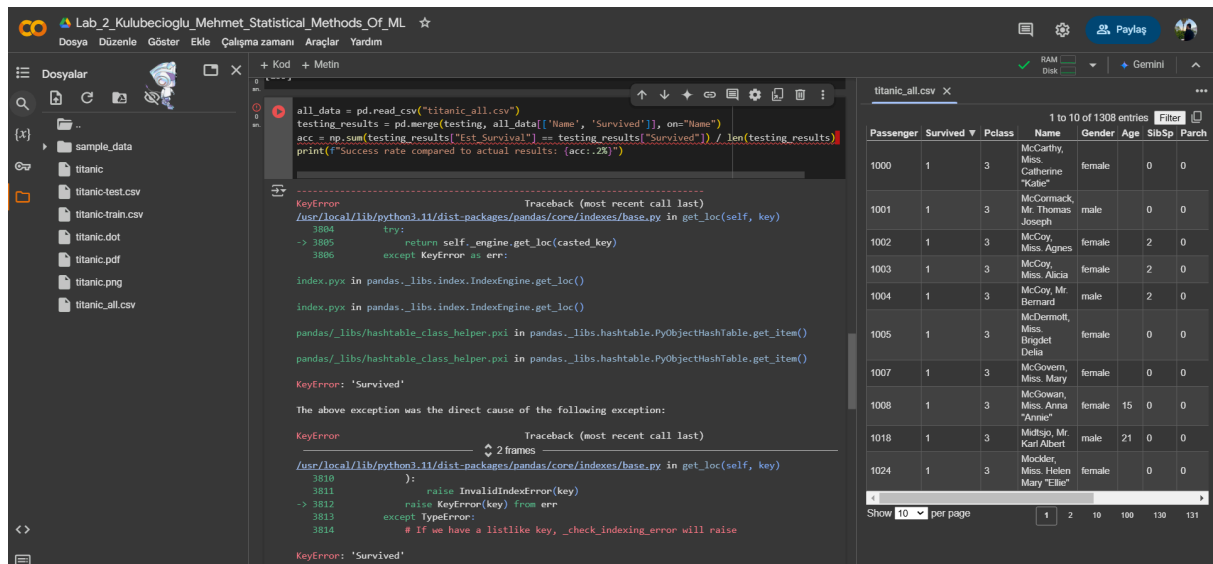
6. Making Predictions on Test Data

After loading the test data (**titanic-test.csv**), the same preprocessing steps were applied, and the model made predictions on whether each passenger survived (**Est_Survival**).



7. Comparison with Real Results

The predictions made on the test dataset were compared with the real survival results (Survived column). The accuracy of the model was computed by comparing predicted values with the actual values.



The screenshot shows a Jupyter Notebook interface with a file explorer on the left, a code editor in the center, and a data preview on the right.

Code Editor:

```
all_data = pd.read_csv("titanic_all.csv")
testing_results = pd.merge(testing, all_data[["Name", "Survived"]], on="Name")
acc = np.sum(testing_results["Est_Survival"] == testing_results["Survived"]) / len(testing_results)
print(f"Success rate compared to actual results: (acc:.2%)")
```

Error Message:

```
KeyError                                Traceback (most recent call last)
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
    3884         try:
-> 3885             return self._engine.get_loc(casted_key)
    3886         except KeyError as err:
KeyError: 'Survived'

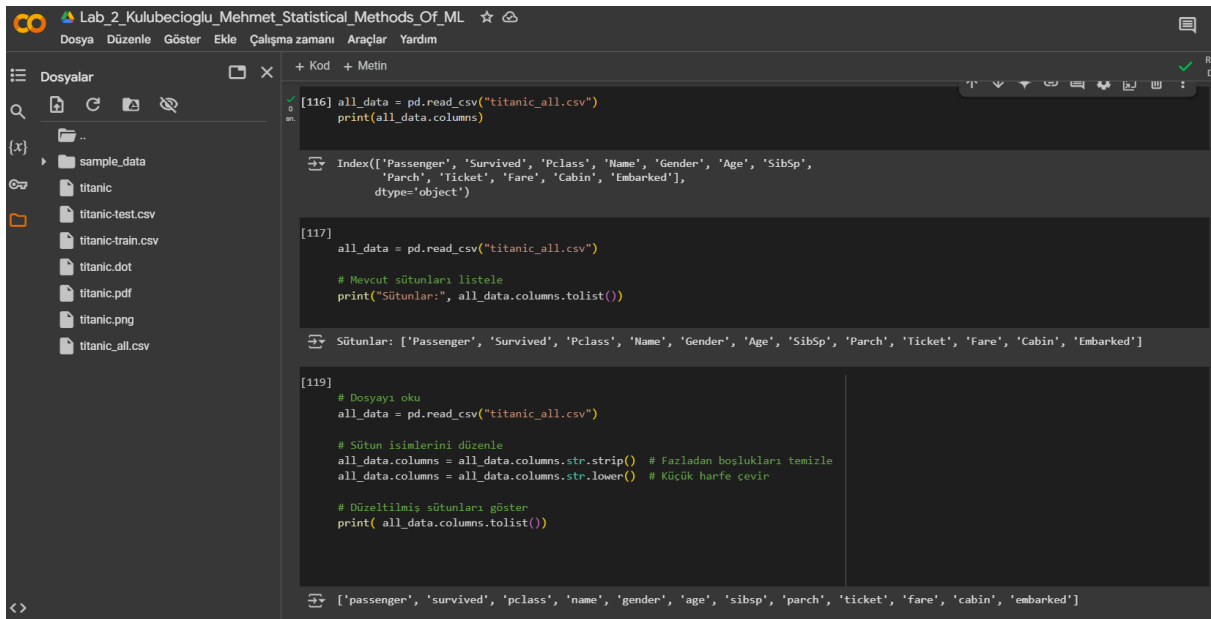
index.pyx in pandas._libs.index.IndexEngine.get_loc()
index.pyx in pandas._libs.index.IndexEngine.get_loc()
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()
pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()
KeyError: 'Survived'

The above exception was the direct cause of the following exception:

KeyError                                Traceback (most recent call last)
2 frames
/usr/local/lib/python3.11/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
    3810         ):
    3811             raise InvalidIndexError(key)
-> 3812         raise KeyError(key) from err
    3813     except TypeError:
    3814         # If we have a listlike key, _check_indexing_error will raise
KeyError: 'Survived'
```

Data Preview (titanic_all.csv):

Passenger	Survived	Pclass	Name	Gender	Age	SibSp	Parch
1000	1	3	McCarthy, Miss Catherine "Katie"	female	0	0	0
1001	1	3	McCormack, Mr. Thomas Joseph	male	0	0	0
1002	1	3	McCoy, Miss. Agnes	female	2	0	0
1003	1	3	McCoy, Miss. Alicia	female	2	0	0
1004	1	3	McCoy, Mr. Bernard	male	2	0	0
1005	1	3	McDermott, Miss. Bridget Della	female	0	0	0
1007	1	3	McGovern, Miss. Mary	female	0	0	0
1008	1	3	McGowan, Miss. Anna "Annie"	female	15	0	0
1018	1	3	Midship, Mr. Karl Albert	male	21	0	0
1024	1	3	Mockler, Miss. Helen Mary "Ellie"	female	0	0	0



```
[116] all_data = pd.read_csv("titanic_all.csv")
      print(all_data.columns)

Index(['Passenger', 'Survived', 'Pclass', 'Name', 'Gender', 'Age', 'SibSp',
      'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')

[117] all_data = pd.read_csv("titanic_all.csv")

# Mevcut sütunları listele
print("Sütunlar:", all_data.columns.tolist())

Sütunlar: ['Passenger', 'Survived', 'Pclass', 'Name', 'Gender', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked']

[119] # Dosyayı oku
all_data = pd.read_csv("titanic_all.csv")

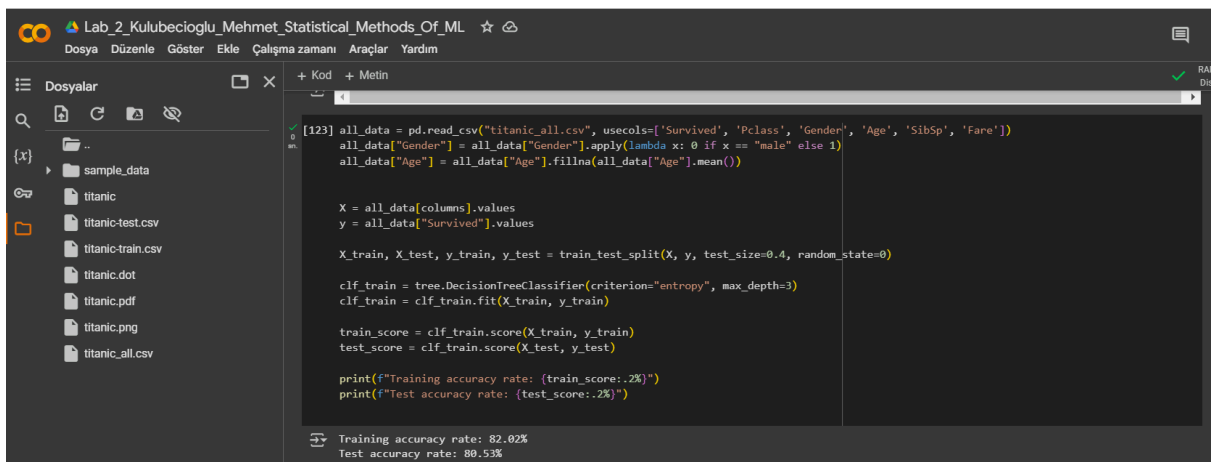
# Sütun isimlerini düzenle
all_data.columns = all_data.columns.str.strip() # Fazladan boşlukları temizle
all_data.columns = all_data.columns.str.lower() # Küçük harfe çevir

# Düzeltilmiş sütunları göster
print(all_data.columns.tolist())

['passenger', 'survived', 'pclass', 'name', 'gender', 'age', 'sibsp', 'parch', 'ticket', 'fare', 'cabin', 'embarked']
```

8. Model Performance

Finally, the accuracy rates on both the training and test datasets were computed. The accuracy on the training dataset was high, while the test dataset accuracy was slightly lower.



```
[123] all_data = pd.read_csv("titanic_all.csv", usecols=['Survived', 'Pclass', 'Gender', 'Age', 'SibSp', 'Fare'])
all_data["Gender"] = all_data["Gender"].apply(lambda x: 0 if x == "male" else 1)
all_data["Age"] = all_data["Age"].fillna(all_data["Age"].mean())

X = all_data[columns].values
y = all_data["Survived"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)

clf_train = tree.DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf_train = clf_train.fit(X_train, y_train)

train_score = clf_train.score(X_train, y_train)
test_score = clf_train.score(X_test, y_test)

print(f"Training accuracy rate: {train_score:.2%}")
print(f"Test accuracy rate: {test_score:.2%}")

Training accuracy rate: 82.02%
Test accuracy rate: 80.53%
```

9. Results and Evaluation

In this assignment, the process of training and testing a decision tree model with the Titanic dataset was successfully completed. The accuracy on the training dataset was high, while the accuracy on the test dataset was somewhat lower. To improve the model's performance, hyperparameter tuning, additional features, or alternative classifiers could be explored.

1. What is the method of decision trees?

The decision tree method is a supervised learning algorithm used for both classification and regression tasks. It creates a model that predicts the value of a target variable based on several input features by learning simple decision rules from the data. The model is represented in a tree structure, where each internal node represents a "test" or "decision" on an attribute, each branch represents the outcome of the test, and each leaf node represents the predicted outcome.

2. Why is the decision tree method used?

The decision tree method is widely used because it is easy to understand, interpret, and visualize. It handles both numerical and categorical data and can model non-linear relationships. Decision trees also perform well with large datasets and are capable of capturing complex patterns in the data. Additionally, they require little data preprocessing, as they can handle missing values and irrelevant features effectively.

3. How are binary trees constructed?

A binary tree is constructed by starting with a root node, and recursively splitting the data into two subgroups based on a decision criterion. Each internal node represents a decision on a feature, and the tree is split into two branches according to the outcomes of that decision. This process continues recursively, forming additional nodes and subtrees until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf, or purity of the nodes).

4. Give an example of using binary trees.

An example of using binary trees is in a loan approval system, where the binary tree helps decide whether to approve a loan based on features such as credit score, income, and debt. Each internal node tests a specific condition (e.g., "Is credit score greater than 650?"), and based on the answer (Yes or No), the tree branches further until it reaches a leaf node that provides the decision (e.g., "Loan Approved" or "Loan Denied").

5. What are the advantages of the decision tree method?

- Easy to interpret and visualize: Decision trees are simple to understand, making them suitable for explaining model decisions.
- Handles both numerical and categorical data: It can be applied to datasets with different types of features.
- No need for extensive data preparation: Decision trees can handle missing values, and feature scaling is not necessary.
- Non-linear relationships: Decision trees can capture non-linear relationships between features.
- Robust to outliers: The algorithm can handle outliers without a significant impact on the model's performance.

6. What are the criteria for splitting the decision tree?

The criteria used to split a decision tree typically include:

- Gini Impurity: Measures how often a randomly chosen element would be incorrectly classified.
- Entropy (Information Gain): Measures the reduction in entropy (or uncertainty) after a split.
- Mean Squared Error (MSE): Used for regression trees to minimize the variance of the target variable in each split.

7. What are the options for stopping decision tree learning?

The learning process of a decision tree can be stopped based on the following conditions:

- Maximum depth: A maximum depth for the tree is specified.
- Minimum samples per leaf: The tree stops growing if a node has fewer than the specified number of samples.
- Minimum impurity decrease: The learning stops if the reduction in impurity is below a threshold.
- Maximum number of nodes: The tree is limited to a specified number of nodes.
- No further improvement: The tree stops growing when no further improvement can be made in the splits.

8. What Python tools are used to build a decision tree?

- **Scikit-learn**: The most common Python library used for building decision trees. It provides the **DecisionTreeClassifier** and **DecisionTreeRegressor** classes.
- **XGBoost**: A powerful and efficient library for building gradient boosting decision trees, often used for structured/tabular data.
- **Graphviz**: A tool for visualizing decision trees in Python using **graphviz** or **pydotplus**.

9. Give an example of implementing a decision tree in Python.

Here's an example of implementing a decision tree using Scikit-learn:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

# Load the dataset
data = pd.read_csv("titanic-train.csv")

# Preprocess the data
data["Age"] = data["Age"].fillna(data["Age"].mean())
data["Gender"] = data["Gender"].apply(lambda x: 0 if x == "male" else 1)

X = data[["Fare", "Pclass", "Gender", "Age", "SibSp"]].values
y = data["Survived"].values

# Split the dataset into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train the decision tree model
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)
clf.fit(X_train, y_train)

# Evaluate the model
accuracy = clf.score(X_test, y_test)
print(f"Accuracy: {accuracy:.2%}")
```

Accuracy: 78.18%

10. How is the evaluation of the decision tree classifier model implemented?

The evaluation of a decision tree classifier is typically done using accuracy, precision, recall, F1-score, and confusion matrix. The most common method is to calculate accuracy, which measures the proportion of correct predictions:

```
from sklearn.metrics import accuracy_score

# Evaluate the accuracy of the model
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2%}")
```

Accuracy: 78.18%

Additionally, metrics such as precision, recall, and F1-score can be computed using **classification_report** from Scikit-learn:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.74	0.97	0.84	164
1	0.92	0.50	0.65	111
accuracy			0.78	275
macro avg	0.83	0.74	0.75	275
weighted avg	0.81	0.78	0.76	275