

**ANL252**

**End-of-Course Assessment**

**H2311509**

**Shafina Binte Hazman**

**3 November 2023**

**Question 1**

The approach chosen is to replace missing values rather than removing them. This decision is driven by several key factors.

First, we will need to identify which data is missing and needs to be replaced to prevent bias from selective removal. With a dataset of this size, each data point carries more weight and is pivotal for making informed decisions and drawing meaningful insights.

import pandas as pd

import numpy as np

# Load the dataset

df\_ECA = pd.read\_csv('ECA.csv')

df\_ECA

PersonID age sex bmi children smoker region charges

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

[1340 rows x 8 columns]

# Count missing values in each column

df\_ECA.isnull().sum(axis = 0/1)

PersonID 0

age 123

sex 0

bmi 0

children 0

smoker 0

region 0

charges 0

dtype: int64

# Check which columns have missing values and assign the result to a new column in df\_ECA

df\_ECA['Columns\_with\_missing\_values'] = df\_ECA.isnull().any()

# Now, access this information within df\_ECA

missing\_columns = df\_ECA['Columns\_with\_missing\_values']

# Check which columns have missing values and print

columns\_with\_missing\_values = df\_ECA.isnull().any()

print(columns\_with\_missing\_values)

PersonID False

age True

sex False

bmi False

children False

smoker False

region False

charges False

Columns\_with\_missing\_values True

dtype: bool

# List the columns with missing values

columns\_with\_missing\_values = df\_ECA.columns[df\_ECA.isnull().any()].tolist()

print(columns\_with\_missing\_values)

['age', 'Columns\_with\_missing\_values']

Next, we standardise the values in the 'sex' column, such as mapping 'F' to 'female' and 'M' to 'male, as it is essential to ensure consistency and uniformity in the dataset. This standardisation simplifies data handling, reduces the risk of errors, and enhances clarity in understanding the data.

# Standardise the 'sex' column values

sex\_mapping = {'F': 'female', 'female': 'female', 'M': 'male', 'male': 'male'}

df\_ECA['sex'] = df\_ECA['sex'].map(sex\_mapping)

# Print the DataFrame to verify the changes

print(df\_ECA)

PersonID age sex bmi children smoker region charges \

0 1 19.0 female 27.900 0 yes southwest 16884.92400

1 2 18.0 male 33.770 1 no southeast 1725.55230

2 3 28.0 male 33.000 3 no southeast 4449.46200

3 4 33.0 male 22.705 0 no northwest 21984.47061

4 5 32.0 male 28.880 0 no northwest 3866.85520

... ... ... ... ... ... ... ... ...

1335 1334 50.0 male 30.970 3 no northwest 10600.54830

1336 1335 18.0 female 31.920 0 no northeast 2205.98080

1337 1336 18.0 female 36.850 0 no southeast 1629.83350

1338 1337 21.0 female 25.800 0 no southwest 2007.94500

1339 1338 61.0 female 29.070 0 yes northwest 29141.36030

Columns\_with\_missing\_values

0 NaN

1 NaN

2 NaN

3 NaN

4 NaN

... ...

1335 NaN

1336 NaN

1337 NaN

1338 NaN

1339 NaN

[1340 rows x 9 columns]

As it remains unaffected by outliers or extreme data points, median imputation is the best method for addressing missing data. When the 'age' variable exhibits skewness or includes outliers, opting for the median as the replacement value is a more fitting decision.

# Calculate the median for the 'age' column

median\_age = df\_ECA['age'].median()

# Print the median age

print("Median Age:", median\_age)

Median Age: 39.0

# Replace missing values in the 'age' column with the median

df\_ECA['age'].fillna(median\_age, inplace=True)

print(df\_ECA['age'])

0 19.0

1 18.0

2 28.0

3 33.0

4 32.0

...

1335 50.0

1336 18.0

1337 18.0

1338 21.0

1339 61.0

Name: age, Length: 1340, dtype: float64

Finally, saving the cleaned dataset in an Excel file serves multiple purposes. It acts as a verification step to confirm the correctness of the data cleaning process. Moreover, it follows best practices by preserving the refined data for future analysis or sharing with collaborators.

# Assuming the code have already run to fill missing values

# Save the DataFrame to an Excel file

df\_ECA.to\_excel('ECA\_Updated.xlsx', index=False)

**Question 2**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load the updated dataset

df\_ECA\_updated = pd.read\_excel('ECA\_Updated.xlsx')

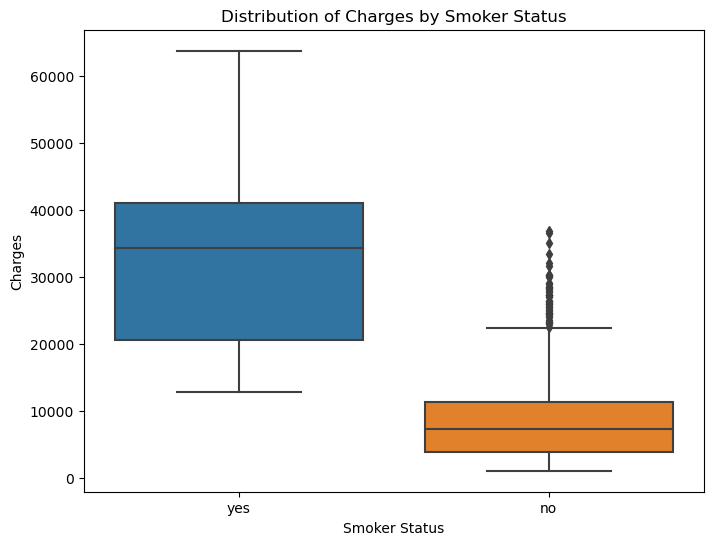


Figure 1: Distribution of Charges by Smoker Status

*This box plot illustrates the contrast in medical charges between smokers and non-smokers in the updated dataset.*

The box plot reveals a clear contrast in medical charges between smokers and non-smokers in the updated dataset. Smokers continue to have notably higher medical charges, as evidenced by the larger median and longer whiskers on their respective box plots.

The interquartile range for smokers is broader, indicating greater variability in charges within this group. Outliers are still primarily concentrated within the smoker category, indicating extreme cases of high charges.

# Figure 2: Age Distribution by Gender (Histogram)

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

sns.histplot(data=df\_ECA\_updated, x='age', hue='sex', kde=True, bins=30)

plt.title('Age Distribution by Gender')

plt.xlabel('Age')

plt.ylabel('Count')

plt.show()

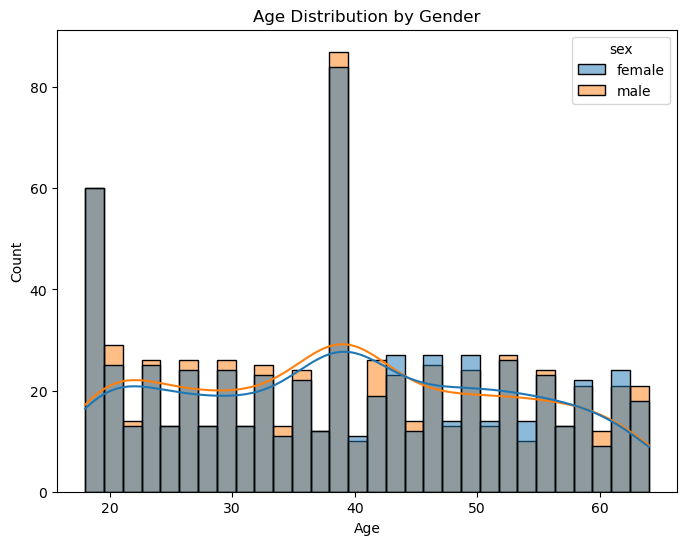


Figure 2: Age Distribution by Gender

*The histogram displays the age distribution by gender in the updated dataset.*

The histogram displays the updated age distribution by gender. It continues to show that the dataset contains similar numbers of males and females, but there is a noticeable difference in age distribution.

Females' age distribution remains slightly right-skewed, with a greater concentration of younger individuals. In contrast, the age distribution for males appears more symmetrical, encompassing a broader range of ages.

# Figure 3: BMI vs. Charges (Scatter Plot)

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

sns.scatterplot(x='bmi', y='charges', data=df\_ECA\_updated, hue='smoker')

plt.title('BMI vs. Charges')

plt.xlabel('BMI')

plt.ylabel('Charges')

plt.show()

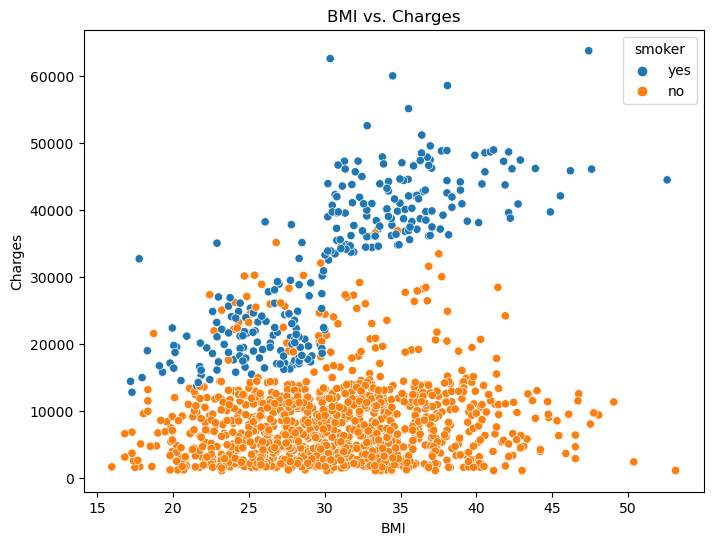


Figure 3: BMI vs. Charges

*This scatterplot highlights the correlation between BMI and medical charges for both smokers and non-smokers.*

The scatter plot continues to reveal a positive correlation between BMI and medical charges, particularly for smokers. Smokers with higher BMI values continue to incur significantly higher charges.

There is also a similar trend among non-smokers, but the correlation isn't as strong. It suggests that BMI remains an important factor in determining medical charges for both groups, with smokers experiencing a greater impact.

Particularly among smokers with high BMI charges, there are outliers who may suffer from specific medical conditions or treatments leading to extremely high costs.

In summary, these visualisations provide valuable insights into the relationships between various factors within the dataset. They highlight the impact of smoking on medical charges, the age distribution by gender, and the correlation between BMI and charges, shedding light on the underlying trends and variations in the data.

**Question 3**

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

from sklearn.tree import plot\_tree

import pandas as pd

We should begin by creating a decision tree classifier. Decision trees are a powerful tool for exploring relationships within data. They break down the decision-making process into a tree-like structure, making it easier to interpret.

# Create a Decision Tree Classifier with a maximum depth of 3

decision\_tree = DecisionTreeClassifier(max\_depth=3, min\_samples\_split=10, min\_samples\_leaf=5, random\_state=42)

Before using the data for training, we are required to perform one-hot encoding on the dataset. This is essential because decision trees work with numerical data, and the 'sex' and 'region' columns contain categorical values. One-hot encoding transforms these categorical variables into binary columns, making them suitable for the model.

# Perform one-hot encoding for 'sex' and 'region' columns

X\_encoded = pd.get\_dummies(df\_ECA\_updated, columns=['sex', 'region'], drop\_first=True)

To evaluate the model's performance, we divided the dataset into two segments: a training set and a testing set. The training set was employed for model training, while the testing set was utilised to evaluate how effectively the model can adapt to unfamiliar data.

# Split the data into features (X) and the target variable (y)

X = X\_encoded.drop('smoker', axis=1) # Features (independent variables)

y = X\_encoded['smoker'] # Target variable

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

The decision tree model is trained on the training data. It learns to make decisions based on the independent variables to predict the 'smoker' status. The model constructs a tree structure where each node represents a decision point.

# Train the Decision Tree model

decision\_tree.fit(X\_train, y\_train)

As a result of training the model, we use it to predict the 'smoker' status of the testing data. The model evaluates the features of each individual in the testing set and assigns a 'smoker' or 'non-smoker' label based on the decision rules learned during training.

# Predict 'smoker' status on the test data

y\_pred = decision\_tree.predict(X\_test)

# Evaluate the model's accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

To assess the model's performance, we need to calculate its accuracy. A model's accuracy is measured by how well its predictions match actual data about smoker status. Models with higher accuracy are more effective.

# Print the accuracy of the model

print(f"Model Accuracy: {accuracy:.2f}")

Model Accuracy: 0.97

In addition to accuracy, we generate a classification report. A more comprehensive assessment is provided in this report, including metrics such as precision, recall, and F1-score for non-smokers and smokers. It helps us understand how well the model distinguishes between the two categories.

# Generate a classification report

class\_report = classification\_report(y\_test, y\_pred, target\_names=['Non-Smoker', 'Smoker'])

print("Classification Report:\n", class\_report)

Classification Report:

precision recall f1-score support

Non-Smoker 1.00 0.96 0.98 207

Smoker 0.88 1.00 0.94 61

accuracy 0.97 268

macro avg 0.94 0.98 0.96 268

weighted avg 0.97 0.97 0.97 268

To gain insights into the model's decision-making process, we visualise the Decision Tree. The tree diagram illustrates the hierarchy of decisions the model uses to classify individuals as smokers or non-smokers. This visual representation is helpful for understanding the factors that influence an individual's smoking status.

# Plot the Decision Tree

plt.figure(figsize=(10, 8))

plot\_tree(decision\_tree, filled=True, feature\_names=X.columns.tolist(), class\_names=['Non-Smoker', 'Smoker'])

plt.title('Decision Tree for "Smoker" Prediction')

plt.show()

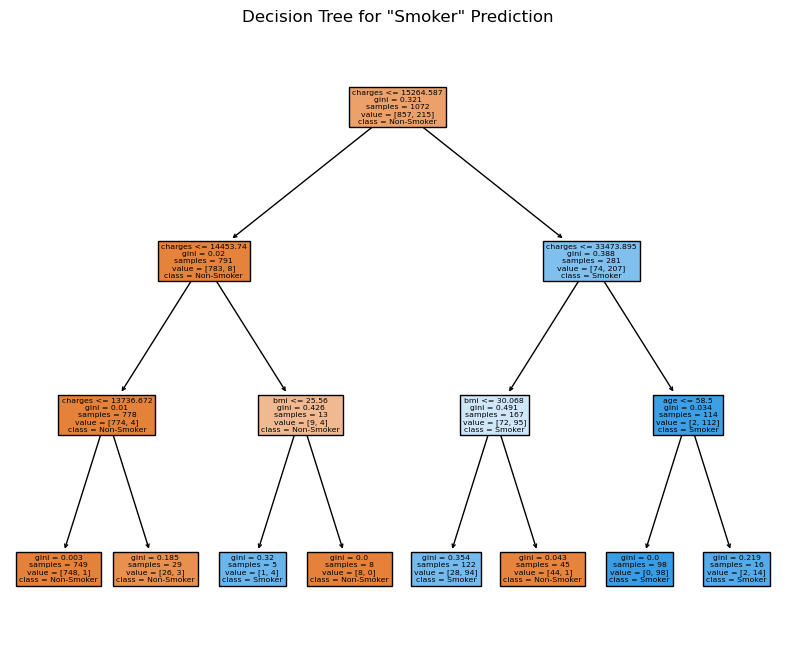


Figure 4: Decision Tree for "Smoker" Prediction

*Visualisation of the decision tree built to predict an individual's "smoker" status. Each node in the tree represents a decision point, and the leaves represent the predicted classes, "Non-Smoker" or "Smoker."*

**Question 4**

The decision tree in Figure 4 illustrates the hierarchical structure employed to predict an individual's "smoker" status based on features such as age, BMI, medical charges, region, and gender. This decision tree offers valuable insights into the decision-making process and the significant factors determining smoking behaviour.

Age as the Primary Split: The first split in the tree occurs based on age, specifically whether an individual is older than 38.5 years. This suggests that age is a critical factor in determining smoking behaviour, with older individuals more likely to be non-smokers.

BMI and Charges: For individuals younger than 38.5 years, the tree further splits based on BMI and medical charges. Higher charges, combined with a BMI greater than 30.2, indicate a higher likelihood of being a smoker.

Non-Smoking Majority: The majority of individuals are categorised as non-smokers, as seen by the larger "Non-Smoker" leaf in the tree's final nodes.

**Question 5**

Feature Importance: The importance of features in a decision tree depends on their position within the tree structure. Analysing exploratory data can identify influential factors in a dataset by assessing feature importance scores. A high importance feature provides valuable insight into the dataset, guiding further analysis or hypothesis formulation (Loh, 2011).

Hierarchy of Decision Rules: A decision tree represents data hierarchically. They reveal patterns and relationships in data by revealing decision paths and rules. Using this hierarchy, analysts and stakeholders can understand and access complex data structures (Mingers,1989).

Multivariate Analysis: Decision trees inherently support multivariate analysis by considering interactions between variables. Exploratory data analysis reveals how variables interact and influence outcomes by exploring branches and nodes. This is vital for detecting hidden relationships within the data (Quinlan, 1993).

Data Partitioning: By segmenting data into subsets, decision trees produce clusters with similar characteristics. The partitions facilitate the identification of subpopulations or segments within the data. Using these segments, analysts can gain a deeper understanding of the data's heterogeneity (Breiman et al., 1984).

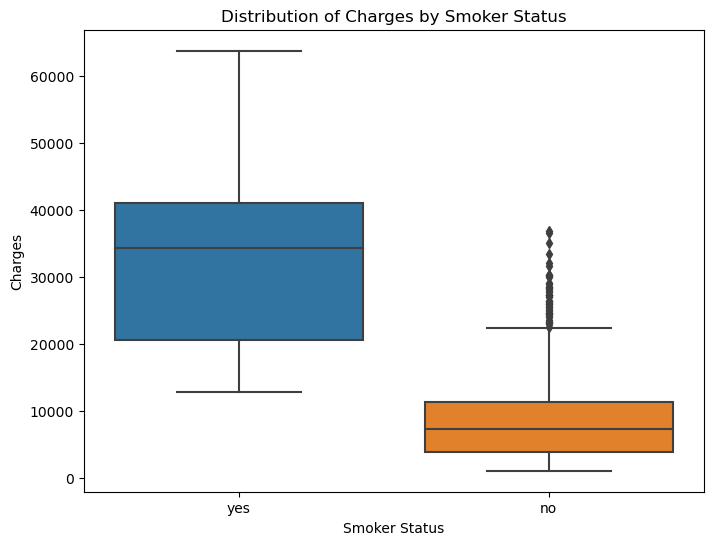
Visualisation: Decision trees provide a highly intuitive visual representation. This allows analysts to communicate findings clearly and effectively. It is possible to present complex data structures in a comprehensible manner by using decision tree visualisations.

As a result, decision trees are valuable tools for exploratory data analysis that go far beyond their traditional predictive capabilities. By providing insights into feature importance, decision hierarchies, interactions, and data segmentation, they enable intuitive visualisations.

By combining these factors, users can understand the importance of features, hierarchy of decision making, interactions, data segmentation, and intuitive visualisation of data

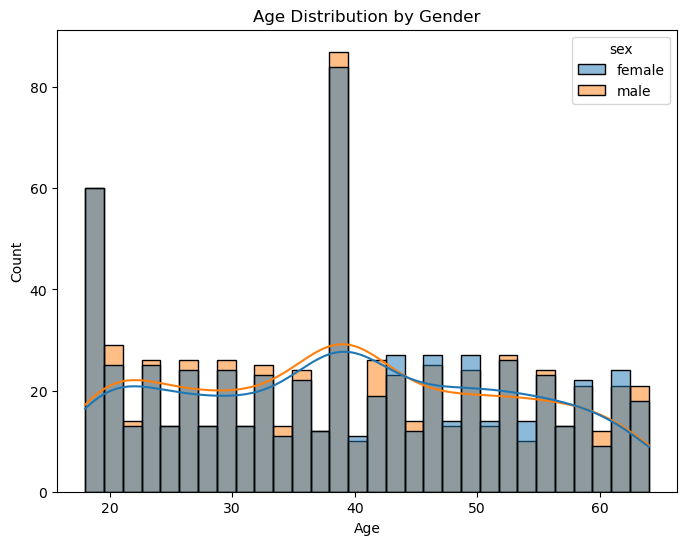
**Appendix A: Supplementary Figures**

Figure 1: Distribution of Charges by Smoker Status



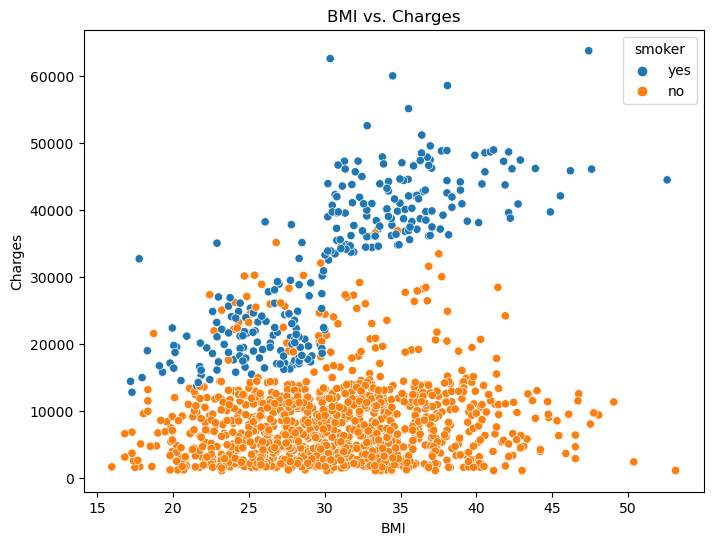
This box plot illustrates the contrast in medical charges between smokers and non-smokers in the updated dataset.

Figure 2: Age Distribution by Gender



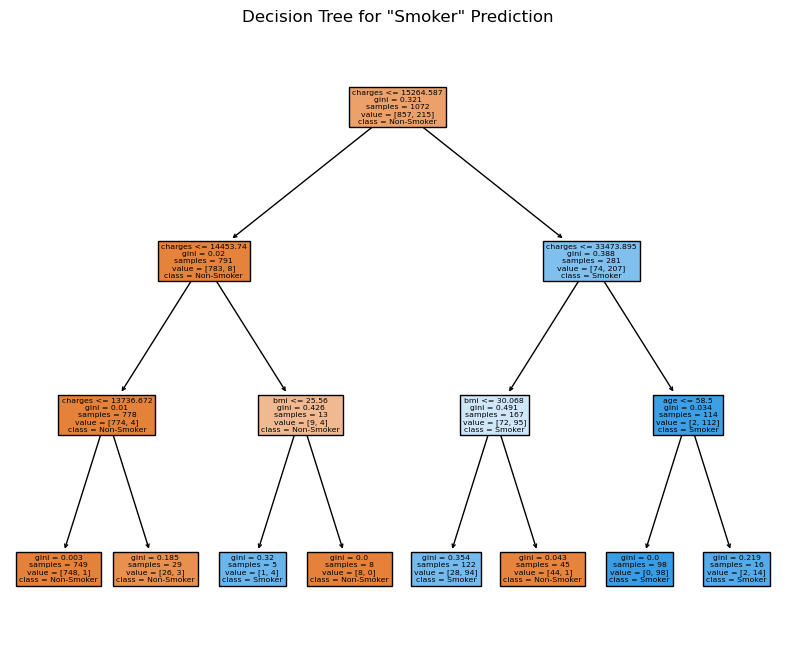
The histogram displays the age distribution by gender in the updated dataset.

Figure 3: BMI vs. Charges



This scatterplot highlights the correlation between BMI and medical charges for both smokers and non-smokers.

Figure 4: Decision Tree for "Smoker" Prediction



Visualisation of the decision tree built to predict an individual's "smoker" status. Each node in the tree represents a decision point, and the leaves represent the predicted classes, "Non-Smoker" or "Smoker."

**References**

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and regression trees. CRC Press.

Loh, W. Y. (2011). Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(1), 14-23.

Mingers, J. (1989). An empirical comparison of selection measures for decision tree induction. Machine Learning, 3(4), 319-342.

Quinlan, J. R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann.