

**ANL252**

**Python for Data Analytics**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**End-of-Course Assessment**

**July 2023 Presentation**

|  |  |
| --- | --- |
| Name | Goh Jia Yi Samuel |
| SUSS PI No | Q2311483 |

Q1)

import pandas as pd

eca = pd.read\_csv("ECA.csv")

charges = eca["charges"]

import numpy as np

# Normalization

normalized\_data = [(x - np.min(charges)) / (np.max(charges) - np.min(charges)) for x in charges]

# Standardization

standardized\_data = [(x - np.mean(charges)) / np.std(charges) for x in charges]

After applying these transformations, the normalized data list will have values scaled between 0 and 1, while the standardized data list will have values cantered around 0 with a standard deviation of 1.

To interpret the standardized data,

Positive z-score: The original value is above the mean

Negative z-score: The original value is below the mean

Z-score of 0: The original value is equal to the mean

This helps with improving readability of the data as when the scales of the variables are vastly different, it can be challenging to compare the coefficients of a linear model meaningfully. Standardized coefficients can provide insights into which features are more influential.

age = eca["age"]

mean\_values = age.mean()

print(mean\_values)

age = age.fillna(mean\_values)

missing\_count = age.isna().sum()

print(missing\_count)

The age column contains missing data. By filling the missing values with the mean of a column is a common technique used when there are no outliers identified in the data, this ensures that the imputed value does not skew the dataset. Missing values are common in datasets and can adversely affect analysis and modelling and can hinder the effectiveness of machine learning models.

bmi = eca["bmi"]

def categorize\_bmi(bmi):

if bmi < 18.5:

return 'Underweight'

elif 18.5 <= bmi < 24.9:

return 'Normal weight'

elif 24.9 <= bmi < 30:

return 'Overweight'

else:

return 'Obese'

bmi\_category = eca['bmi'].apply(categorize\_bmi)

Body Mass Index (BMI) is a measure of body fat based on height and weight. Although it's numerical, for analysis purposes, it can be more insightful to categorize BMI into standard categories such as 'Underweight', 'Normal weight', 'Overweight', and 'Obese'. This can help in drawing more intuitive insights from the data.

Q2)

import seaborn as sns

import matplotlib.pyplot as plt

# Create a boxplot for smokers

sns.boxplot(x='smoker', y='charges', data=eca[eca['smoker'] == 1])

plt.title("Medical Charges for Smokers")

plt.show()

# Create a boxplot for non-smokers

sns.boxplot(x='smoker', y='charges', data=eca[eca['smoker'] == 0])

plt.title("Medical Charges for Non-Smokers")

plt.show()

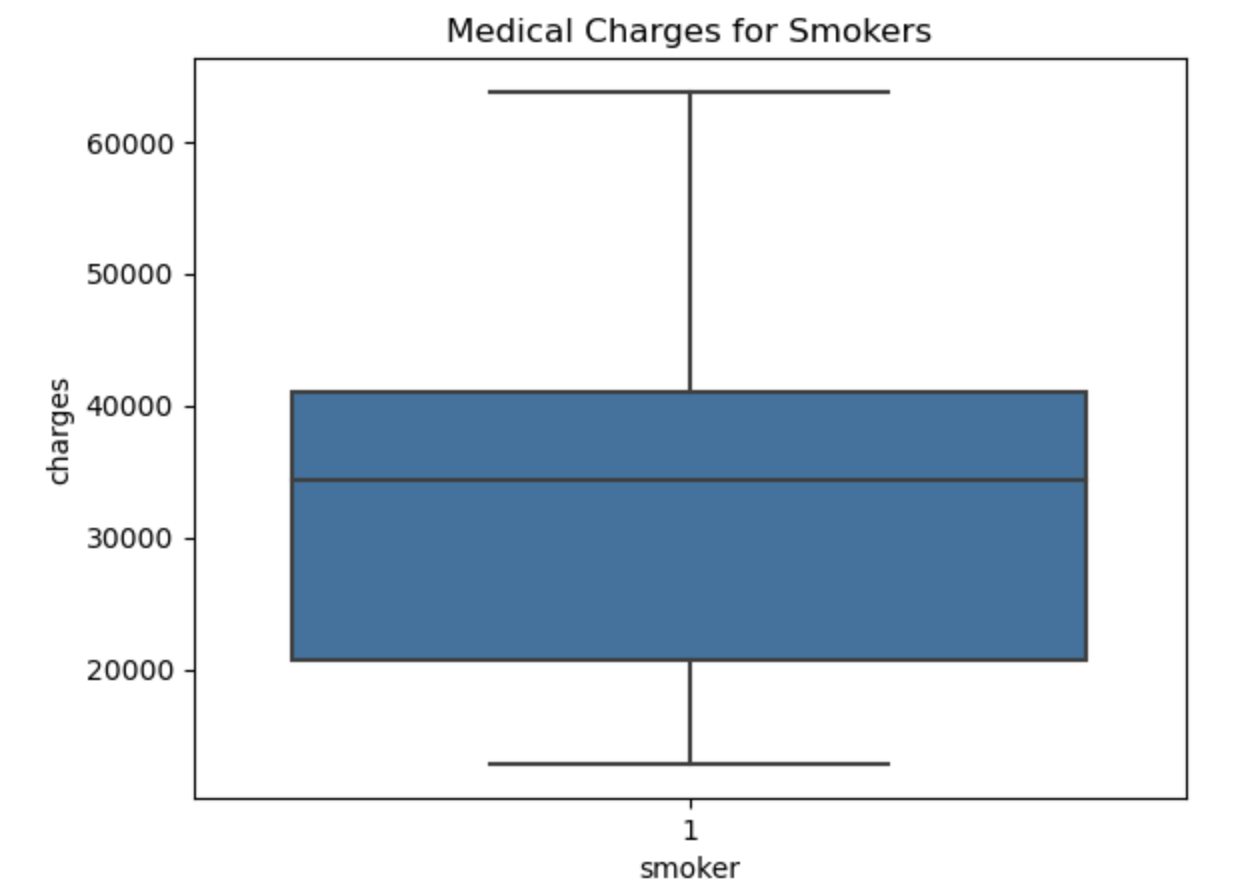
This plot lets us compare the medical charges between smokers and non-smokers.

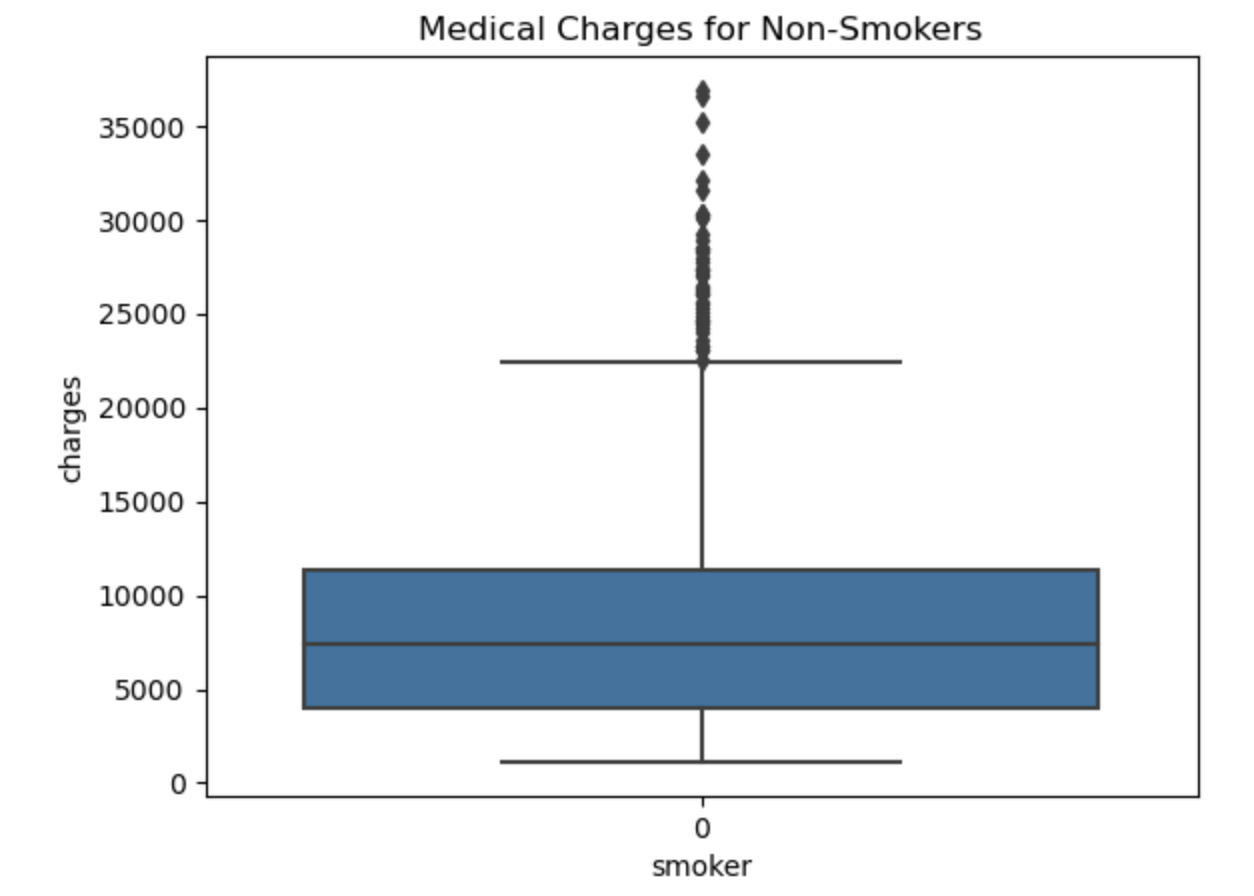
The boxplot can also help identify outliers in each category and give us a sense of the spread and distribution of charges for each group.

Typically, smokers might have higher medical expenses due to diseases or conditions associated with smoking.

From the boxplot, it can be seen that there are outliers within the non-smokers data. Hence, the median of 7345 is a better indicator. However, there is no outliers within the smokers data which allows us to use the mean of 31933 as an indicator.

This provides evidence supporting the fact that smoking might be leading to higher medical costs.





sns.scatterplot(x='age', y='charges', hue='smoker', data=eca)

plt.title('Age vs. Charges colored by Smoker Status')

plt.xlabel('Age')

plt.ylabel('Charges')

plt.legend(title='Smoker')

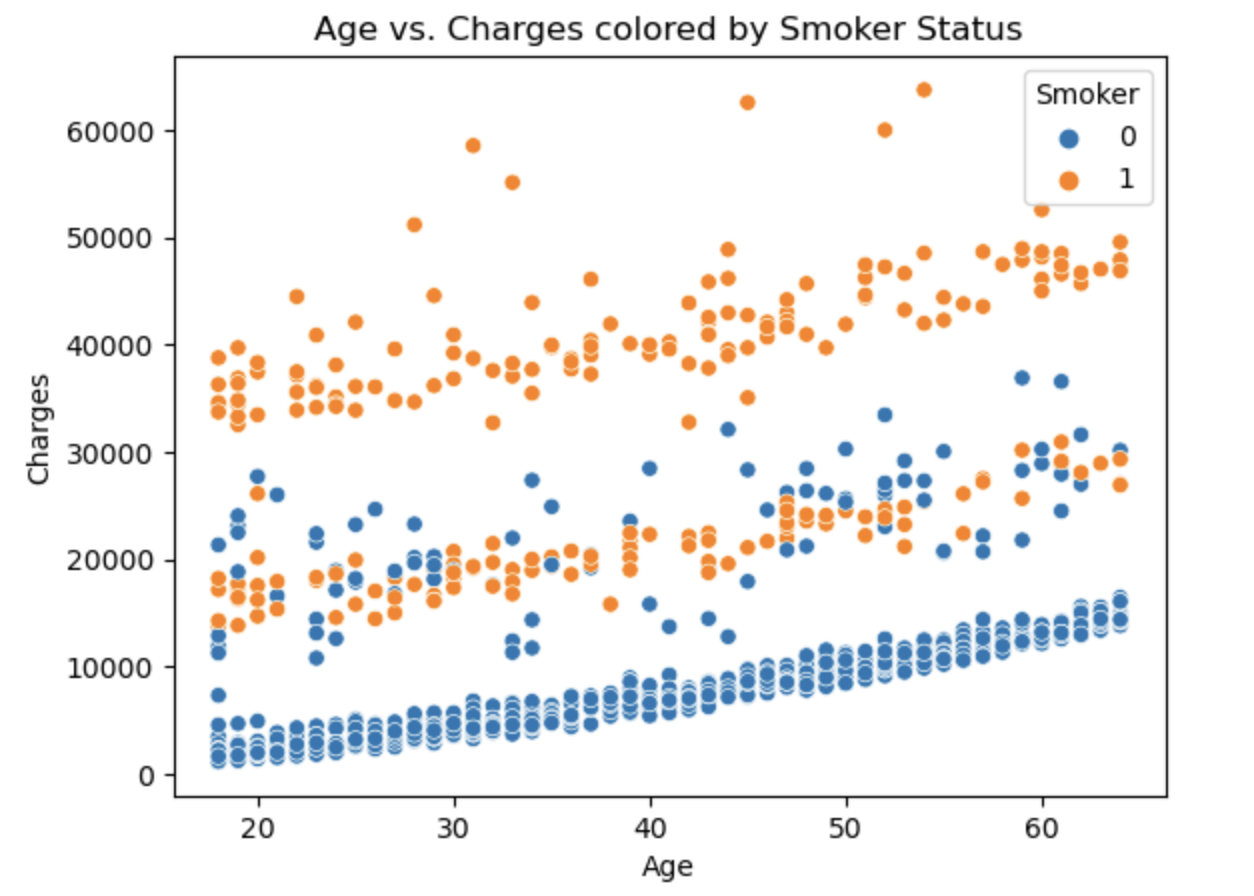
plt.show()

This plot helps understand the relationship between age and medical charges while also considering whether a person smokes.

As age increases, medical charges might generally rise due to increased risk of health issues with aging.

The colour distinction will help understand if smokers, regardless of their age, have higher charges compared to non-smokers of the same age.

Since we see a clear distinction in charges between smokers and non-smokers across all age groups, it reinforces the idea that smoking significantly impacts medical costs. There is also positive relationship between them suggesting that age is correlated to medical cost.



import numpy as np

import matplotlib.pyplot as plt

# Create a histogram

plt.hist(charges, bins=10, edgecolor='black', alpha=0.7)

# Customize the plot

plt.title('Histogram Example')

plt.xlabel('Value')

plt.ylabel('Frequency')

# Show the histogram

plt.show()

Histograms provide a quick and easy way to visualize the distribution of data. They help you see the shape of the data, such as whether it's symmetrical, skewed. In this case, it is a right-skewed histogram. It means that the majority of your data points are concentrated on the left side of the distribution, with a tail extending to the right. This shows that most patients have relatively low healthcare expenses.

A graph of a number of blue bars

Description automatically generated with medium confidence

Q3)

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier, plot\_tree

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

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

# Load the dataset

eca = pd.read\_csv("ECA.csv")

# Impute missing values in the 'age' column with the mean

age\_mean = eca['age'].mean()

eca['age'].fillna(age\_mean, inplace=True)

# Preprocess the data by one-hot encoding categorical variables

eca = pd.get\_dummies(eca, columns=['sex', 'region'], drop\_first=True)

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

X = eca.drop('smoker', axis=1)

y = eca['smoker']

# Split the dataset 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)

# Create and train a decision tree classifier

clf = DecisionTreeClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# Evaluate the model's performance

y\_pred = clf.predict(X\_test)

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

classification\_rep = classification\_report(y\_test, y\_pred)

print("Accuracy:", accuracy)

print(classification\_rep)

# Plot and visualize the Decision Tree

plt.figure(figsize=(25, 15))

plot\_tree(clf, feature\_names=X.columns, class\_names=['no', 'yes'], filled=True, fontsize=8)

plt.savefig('decision\_tree\_plot.png', bbox\_inches='tight')

plt.show()

In this analysis, a Decision Tree was utilized to gain insights into the provided dataset, focusing on the prediction of the 'smoker' variable.

The dataset was loaded using Pandas, and it was observed that the 'age' column contained missing values. To ensure data completeness, the missing values were imputed by replacing them with the mean age value.

Categorical variables, 'sex' and 'region,' were transformed into a numerical format through one-hot encoding. This transformation was essential to enable the Decision Tree classifier to work with the data effectively.

The dataset was partitioned into two subsets, separating features (X) from the target variable (y). 'smoker' was chosen as the target variable for prediction.

To evaluate the model's performance, the dataset was further divided into training and testing sets using an 80/20 split ratio, ensuring the model's ability to generalize.

A Decision Tree classifier was employed for the prediction task, with a focus on interpretability and capturing complex feature relationships. The classifier was trained using the training dataset.

The model's performance was evaluated using the testing dataset. An accuracy score of approximately 98.13% was achieved, indicating the model's proficiency in distinguishing between smokers and non-smokers. Additionally, precision, recall, and F1-score metrics were calculated for both 'smoker' classes ('yes' and 'no').

To gain a visual understanding of the Decision Tree's structure and decision-making process, the tree was visualized and saved as an image. Adjustments were made to the figure size and font size to ensure the readability of the tree structure.

In conclusion, this analysis showcases how a Decision Tree can be effectively employed to explore a dataset, predict 'smoker' status based on various features, and provide valuable insights into the dataset's underlying patterns. The pre-processing steps, handling of missing values, and subsequent model evaluation collectively contribute to the successful application of the Decision Tree classifier in this context.

Q4)

A screenshot of a computer screen

Description automatically generated

A diagram of a company

Description automatically generated

Insights:

The model has an overall accuracy of approximately 98.13%, which indicates it made correct predictions for almost 98 out of every 100 samples in the test set. This signifies a strong performance.

Excellent Precision for "No" Class, the model predicted the "no" class with perfect precision, meaning that every instance it labelled as "no" was indeed a "no".

Strong recall for both classes, recall values indicate how many of the actual positives were correctly identified by the model. Both classes have a recall around 0.98, showing that the model effectively captured the majority of positive instances for both categories.

The F1-score, which is a harmonic mean of precision and recall, is near perfect for the "no" class and excellent for the "yes" class. This demonstrates that the model maintains a good balance between precision and recall.

Q5)

Decision trees can be an invaluable tool for exploratory data analysis beyond their primary use in predictive modelling.

Decision trees provide an intuitive graphical representation of the data, helping to understand the hierarchical relationship between variables. They effectively delineate how different features contribute to outcomes or splits, thus providing insight into the structure of the data.

By observing the order of splits and the nodes' depth, one can infer the importance of different features. Features at the top or closer to the root are generally more significant in determining the target variable than those deeper down the tree.

Decision trees can reveal interaction effects between variables. If two features often split closely together in a tree, they may have a combined effect on the target variable. Recognizing such interactions can be invaluable for further statistical analyses or feature engineering.

Decision trees can help identify anomalies or potential errors in the data. Features that shouldn't logically contribute to a decision but do might hint at data quality issues.

Decision trees naturally handle non-linear relationships, allowing for a nuanced understanding of complex datasets where linear assumptions don't hold.

The terminal leaves of a decision tree segment the data into homogeneous groups. Analysing these groups separately might provide insights into distinct patterns within the dataset.

However, while using decision trees for EDA, it's essential to be aware of their limitations. They are prone to overfitting, especially with small datasets or without proper pruning. Hence, the insights derived should be corroborated with other analytical methods for robustness.

In conclusion, decision trees, traditionally used for predictions, offer a structured and visual approach for exploratory data analysis, highlighting key features, interactions, and patterns in the data (Han et al., 2011).

Reference:

Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.