

**ANL252**

**Python for Data Analytics**



**End-of-Course Assessment**

**July 2023 Presentation**

NAME: NUR AMEERAH BINTE NASSIRUDIN

SUSS PI NO: Y2170573

SUBMISSION DATE: 03 NOVEMBER 2023

**Question 1)**

import pandas as pd

import numpy as np

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

eca\_df = df.dropna()

print(eca\_df.to\_string())

# Using nested replace()

# Replace multiple characters at once

res = eca\_df.replace('female', 'F').replace('male', 'M').replace('yes', 'Yes').replace('no', 'No').replace('northwest', 'North-West').replace('northeast', 'North-East').replace('southwest', 'South-West').replace('southeast', 'South-East')

​

#recode smoker into binary

res["smoker"].replace({'Yes': 1, 'No': 0}, inplace=True)

res["sex"].replace({'F': 1, 'M': 0}, inplace=True)

​

res["region"].replace({'South-West': 1, 'South-East': 2, 'North-West': 3, 'North-East': 4}, inplace=True)

​

print(res)

We will be able to create charts that will allow us to easily view and understand the data by using nested replace for specific data. For example, we could standardize the dataset by changing "female" to F and "male" to M. This would simplify the charting process as we go on.

# Drop Mean +/- 3SD

mean\_minus\_3sd = res["charges"].mean() - ( res["charges"].std() \* 3 )

mean\_plus\_3sd = res["charges"].mean() + ( res["charges"].std() \* 3 )

print(mean\_minus\_3sd)

print(mean\_plus\_3sd)

3SD cleaning is a data cleaning or data preprocessing technique that involves removing outliers based on a 3-times-the-standard deviation (3SD) threshold. After running the code above, the outliers in the dataset will be removed using the 3SD rule. Outliers can cause statistical analyses and data visualizations to be distorted, so removing them can lead to more accurate and reliable results. By removing extreme outliers, the data may more closely resemble a normal distribution, which is frequently an underlying assumption in many statistical methods. This can lead to more accurate inferences and improved model performance.

res["charges"] = res["charges"].map(lambda x : x if (x >= mean\_minus\_3sd and x <= mean\_plus\_3sd) else np.NaN)

res["chargesZ"] = np.log(res['charges']) #Transform using natural Log

res["chargesZ"] =( res["chargesZ"] - res["chargesZ"].mean() ) / res["chargesZ"].std() #calculate the Z score

In data analysis, a natural logarithm (ln) transformation is frequently used to convert data that is not normally distributed into a more normally distributed form. The ln transformation can reduce the impact of outliers in data, making statistical analyses more robust to the presence of such outliers. This has the potential to be useful for statistical analysis and modeling.

**Question 2)**

res.hist()

array([[<Axes: title={'center': 'PersonID'}>,

<Axes: title={'center': 'age'}>],

[<Axes: title={'center': 'bmi'}>,

<Axes: title={'center': 'children'}>],

[<Axes: title={'center': 'charges'}>, <Axes: >]], dtype=object)

A graph of different sizes and numbers

Description automatically generated with medium confidence

We can see from the histogram that there are more than 500 females and males. Males are represented by the number '0,' and females by the number '1'. However, based on the histogram above, we can see that there are slightly more males than females in the dataset.

Following that, we can see that the region is more or less even, with the South-East region having the most people, as represented by the value '2' in the chart above. Finally, as shown in the graph above, the number of smokers is less than 500. This is due to the fact that smokers are represented by the value '1', whereas non-smokers are represented by the value '0'.

**Question 3)**

**Question 4)**

Decision trees are extremely sensitive to distorted data, and even minor changes in the dataset can result in a different decision tree. However, this can be reduced by bagging and boosting algorithms.

Because decision trees are biased when the dataset is imbalanced, it is recommended that the dataset be balanced before creating the decision tree.

**Question 5)**

Decision trees can provide insight into the significance of various features in a dataset. By examining the structure of the tree, decision trees can help identify which features are used for splitting and thus gain an understanding of which variables have the most influence on the target variable. This can be useful for feature selection as well as understanding the driving factors in the dataset.

Decision trees can also be used to illustrate how data is partitioned based on feature values. These visualizations can aid in exploring the data's hierarchical structure and revealing patterns and relationships. The tree can be used to provide a clear and interpretable representation of the data's structure.

Decision trees can be used to test assumptions about data relationships. For example, we can see if the tree structure corresponds to our domain knowledge or expectations about how variables should influence the target variable.

Decision trees can detect data anomalies that may indicate data quality issues. If unusual splits or patterns are observed, this could indicate that there are issues with the data.

Decision trees, on the other hand, are prone to overfitting, particularly when they are deep and complex. Overfitting occurs when the tree detects noise or specific patterns in the training data that do not generalize well to previously unseen data. This can result in poor forecasting performance. Furthermore, decision trees are simple models that may not capture complex data relationships as well as other algorithms. They may struggle with problems requiring sophisticated feature engineering or involving subtle interactions between features.

While decision trees are a useful tool for exploratory data analysis, it is important to keep in mind that they have limitations such as overfitting, sensitivity to small changes in the data, and difficulty capturing some complex relationships.

**References:**

1. Jajodia, P. (12 September 2022), Removing Outliers Using Standard Deviation in Python, KD Nuggets

<https://www.kdnuggets.com/2017/02/removing-outliers-standard-deviation-python.html>

1. Alex (13 June 2023), Python Natural Log: Quick Guide for Calculations, Enterprise DNA Blog

<https://blog.enterprisedna.co/python-natural-log/#:~:text=To%20compute%20the%20natural%20logarithm,from%20the%20Python%20NumPy%20library>.

1. (February 2023), Decision Tree Classification in Python Tutorial, DataCamp

<https://www.datacamp.com/tutorial/decision-tree-classification-python>

1. N.d, 8 Key Advantages and Disadvantages of Decision Trees, Inside Learning Machines

<https://insidelearningmachines.com/advantages_and_disadvantages_of_decision_trees/>