A red and blue logo

Description automatically generated

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**End-Course Assignment (ECA) July 2023**

|  |  |
| --- | --- |
| **Name** | **TAN ZEKAI MARCUS** |
| **SUSS PI No** | **H2011558** |
| **Tutorial Group** | **T03** |
| **Course Code** | **ANL252\_PYTHON\_FOR\_DATA\_ANALYTICS** |
| **Title of TMA** | **ANL252\_ECA\_Marcustan015\_TanZekaiMarcus** |
| **Submission Date** | 02/11/2023 |

|  |
| --- |
| 1)  First, we will use pandas to load the csv file in python and display the shape to make sure it was imported properly.  import pandas as pd  import numpy as np  ECA\_DATA = pd.read\_csv('ECA.csv')  ECA\_DATA.shape  Output:  (1340, 8)   1. For the first data pre-processing task, we will check for missing values in the dataset using the code below.   missing\_values = ECA\_DATA.isnull().sum()  missing\_values  The code above will produce the output below.  PersonID 0  age 123  sex 0  bmi 0  children 0  smoker 0  region 0  charges 0  dtype: int64  This shows that there are 123 missing values from the age column. However, since there are only 123 missing values out of 1340 of them (9.18%). We shall remove the roles to main data accuracy and clean the data.  cleaned\_data = ECA\_DATA.dropna(subset=['age'])  cleaned\_data.shape  Output:  (1217,8)   1. Next, we will find out if there are any duplicated rows and what row number they belong to using the code below.   duplicated\_rows = eca\_data[eca\_data.duplicated(keep=False)]  duplicated\_rows  Output:    Output shows that rows 100 and 101 are duplicates of row 99. We shall remove these rows using the drop\_duplicates() function.  drop\_duplicates = cleaned\_data.drop\_duplicates()  drop\_duplicates.duplicated().sum()  Output:  0   1. Using the code below, we can see that some of the values in ‘sex’ column are not standardize correctly.   sex\_values = drop\_duplicates['sex'].unique()  sex\_values  Output:  array(['female', 'male', 'F', 'M'], dtype=object)  We will replace all the ‘M’ and ‘F’ to ‘Male’ and ‘Female’ respectively for data cleaning and standardization.  drop\_duplicates['sex'] = drop\_duplicates['sex'].replace({‘M’: 'male', ‘F’: 'female'})  unique\_sex\_values = drop\_duplicates['sex'].unique()  unique\_sex\_values  Output:  array(['female', 'male'], dtype=object) |

|  |
| --- |
| 2a)   1. **Scatterplot of Medical Charges vs. Age by Smoking Status**   import matplotlib.pyplot as plt  plt.figure(figsize=(12, 8))  ax = plt.gca()  smokers = drop\_duplicates[drop\_duplicates['smoker'] == 'yes']  non\_smokers = drop\_duplicates[drop\_duplicates['smoker'] == 'no']  ax.scatter(smokers['age'], smokers['charges'], color='orange', alpha=0.6, label='Smoker', s=50, marker='x')  ax.scatter(non\_smokers['age'], non\_smokers['charges'], color='blue', alpha=0.6, label='Non-Smoker', s=50, marker='x')  plt.title('Medical Charges vs. Age by Smoking Status')  plt.xlabel('Age')  plt.ylabel('Charges')  plt.grid(True, which='both', linestyle='--', linewidth=0.5)  plt.legend()  plt.show()    **Justification:**  The goal of this scatterplot is to compare the medical charges amount incurred base on patient’s age and smoking status. From the chart above, we can observe that most non-smoking patients have a medical charge of below $20000, although the price does gradually with age likely due to the health complications that comes with it. While most smoking patients have a medical charge of above $20000, with all the outliers in the above $50000 range belonging to smokers exclusively. This data therefore shows the potential financial burden smoking can cause for healthcare expenses across different age groups.   1. **Box Plot of BMI by Sex**   male\_bmi = drop\_duplicates[drop\_duplicates['sex'] == 'male']['bmi']  female\_bmi = drop\_duplicates[drop\_duplicates['sex'] == 'female']['bmi']  plt.figure(figsize=(10, 6))  ax = plt.gca()  ax.boxplot([male\_bmi, female\_bmi], vert=True, patch\_artist=True, labels=['male', 'female'], widths=0.7)  plt.title('Box plot of BMI by Sex')  plt.ylabel('BMI')  plt.grid(True, which='both', linestyle='--', linewidth=0.5, axis='y')  plt.show()    **Justification:**  The goal is to compare the summary of the BMI range against the two different Sexes in the US. By observation we can see that the median, interquartile range, minimum and maximum range for male BMI is slightly higher than female, we can also observe that the outliers for male BMI are much higher compared to females, with some reaching the above 50 BMI range. In summary, while both females and males have relatively similar BMI distributions, the slight variation and higher values for male can suggest unhealthier eating habits.   1. **Distribution of BMI Categories**   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 'Obesity'  drop\_duplicates['bmi\_category'] = drop\_duplicates['bmi'].apply(categorize\_bmi)  bmi\_category\_counts = drop\_duplicates['bmi\_category'].value\_counts()  plt.figure(figsize=(10, 8))  bmi\_category\_counts.plot.pie(autopct='%1.1f%%', startangle=90)  plt.title('Distribution of BMI Categories')  plt.ylabel('') # Removing the default 'bmi\_category' label on y-axis for clarity  plt.show()    **Justification:**  Using Pie Chart, we categorize every patient’s BMI using the World Health Organization (WHO) standard classifications to group them into a weight category (WHO,2021). Upon observation of the pie chart, most patients are Obese and if we include those who are overweight, a supermajority of patients is at the risk of impaired health consequences which may lead to disability or even death. This could be implications of the high medical charges observed in the scatterplot and issues with dietary/nutritional accessibility in the USA.  https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight |

|  |
| --- |
| 3) from sklearn.model\_selection import train\_test\_split  from sklearn.tree import DecisionTreeClassifier  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  from sklearn.preprocessing import LabelEncoder  from sklearn.tree import plot\_tree  # Convert categorical variables into numerical format using label encoding  label\_encoders = {}  for column in ['sex', 'region']:  le = LabelEncoder()  drop\_duplicates[column] = le.fit\_transform(drop\_duplicates[column])  label\_encoders[column] = le  # Splitting the dataset into training and testing sets  X = drop\_duplicates[['age', 'sex', 'bmi', 'children', 'region']]  y = drop\_duplicates['smoker']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  # Building the Decision Tree Classifier  clf = DecisionTreeClassifier(max\_depth=3, random\_state=42)  clf.fit(X\_train, y\_train)  # Predictions  y\_pred = clf.predict(X\_test)  # Evaluation  accuracy = accuracy\_score(y\_test, y\_pred)  classification\_rep = classification\_report(y\_test, y\_pred)  conf\_matrix = confusion\_matrix(y\_test, y\_pred)  print("Accuracy:")  print("-----------------")  print(f"{accuracy:.4f}")  print("\nClassification Report:")  print("-----------------")  print(classification\_rep)  print("\nConfusion Matrix:")  print("-----------------")  print(conf\_matrix)  Output:  Accuracy:  -----------------  0.7726  Classification Report:  -----------------  precision recall f1-score support  no 0.78 0.99 0.87 283  yes 0.40 0.02 0.05 82  accuracy 0.77 365  macro avg 0.59 0.51 0.46 365  weighted avg 0.69 0.77 0.69 365  Confusion Matrix:  -----------------  [[280 3]  [ 80 2]]  We will use the following code to create our decision tree to further explore the dataset, the code programmed, and decision taken is as follows:   1. Import the necessary libraries from sklearn such as building decision tree classifiers, plotting of the tree and model evaluation. 2. Since ‘Sex’ and ‘Region’ are strings values, they need to be converted into numerical format for the decision tree via label encoding, their values are then stored in the label\_encoder dictionary which will be use later to understand the mapping. 3. The variable X will contain all the dependent variables while Y will contain the independent variable (‘Smoker’), the data will then be split into training and testing. 4. The decision tree classifier is then build using the training data, with a maximum tree row of 3. 5. Variable y\_pred is created to predict whether a person is a ‘smoker’ based on the trained decision tree model using decision according to the dependent variables. 6. We will then evaluate the model’s performance using the accuracy\_score, classification\_report and confusion\_matrix metrics. Based on the output, we can determine the following results below:   **Accuracy:**   * Prediction accuracy of smoker status – 77%   **Classification Report:**   * Number of all smokers identified (0.02%) * Number of all non-smokers identified (99%) * True-negative Number of smokers correctly predicted as smokers (2, recall- 40%) * True-positives Number of non-smokers correctly predicted as non-smokers (280, recall - 78%) * False Positive Number of non-smokers incorrectly predicted as smokers (3) * False Negative Number of smokers incorrectly predicted as non-smokers (80) |
| 4) # Setting up the figure size  plt.figure(figsize=(20, 12))  # Plotting the decision tree  plot\_tree(clf, filled=True, feature\_names=X.columns, class\_names=clf.classes\_, rounded=True)  plt.title("Decision Tree for Predicting Smoking Status")  plt.show()    When observing the decision tree, we can gather that the matrix bases its primary decision on ‘Sex’ in the root node. If true the tree will continue on the left node, and right for false. In the label encoder, female is defined as 0 while male is defined as 1. Therefore **sex <=0.5** indicates if sex is female as a primary decision. On the left side of the 2nd branch it further segments it’s data based on BMI value, the tree decides that the condition **bmi <= 21.87** is important in identifying smoking patterns. If true the 3rd branch will then identify the region the patient belongs in as a deciding factor. Based on the label encoder, **region <=0.5** indicates that the patient is living in the northeast. Therefore, if true it will finally reach the grey leaf node where the **class = yes** indicating that the patients in this node are majority smokers, as the indication of the **values = [7,8]** shows that 8 of the patients in this node are smokers and 7 of them are not. |

|  |
| --- |
| 5) While decision trees are mainly used for predictive modelling, they can also be used as valuable tools for exploratory data analysis (EDA).  When visualizing the decision tree, we can see that it splits the data into different note based on featured values, this allows the user to interpret how each values influence the outcome of the tree and provide insights into the relationship between them.  We can also determine the significant of each value base on where they rank in the tree nodes, with the root node of the tree being the most influential value for prediction.  Linear regression models are also used frequently in EDA, but the issue is that they might miss out on non-linear relationships. However, decision trees can capture them and identify unexpected patterns.    Decision trees can also handle both numerical and categorical data which lessen the time taken for analysis compared to traditional EDA methods where extensive pre-processing of data ins involved.  One example of a real-life application is that decision trees is used by healthcare data scientist to analyse a patient’s medical history to determine the potential of contracting certain diseases (Niklas Donge, 2023). This example is very similar to our decision tree in question 4 where we analyse the various factors that highly contribute a patient to be a smoker. Therefore, by analysing these factors/value, we can draw a few correlations to the independent variable input like traditional EDA methods.  (Niklas Donge, 2023) Random Forest: A Complete Guide for Machine Learning- https://builtin.com/data-science/random-forest-algorithm  (WHO,2021) Obesity and overweight - https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight |