A red and blue text on a black background

Description automatically generated

**ANL252**

**Python for Data Analytics**

**End of Course Assignment**

**Jul 2023**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI Number** |
| Gwendolyn Yeo Zi Hui | Y2310224 |

**Tutorial Group**: T03

**Instructor’s Name**: Professor Kumar Munish

**Submission Date**: 03/11/2023

**Question 1**

For Python to read the dataset, the following code will be used:

|  |
| --- |
| **CODE**  import pandas as pd  # Read file  df = pd.read\_csv('ECA.csv')  # To find out the total number of rows & columns  df.shape  **Output:** (1340,8)  Based on the output above, we will know that the data file “ECA” has **1340 rows and 8 columns.** |

Pre-Processing Task 1 – Handle Missing Value

For Python to identify missing values & replace them, I will use

* isnull().sum() to find out how many null values does each individual column has
* df.fillna(value=df["age"].mean().astype(int)) to replace missing values with the mean of respective column.

When evaluating medical cost data, it is crucial to find missing values and substitute them with the mean. This preserves the dataset's validity and facilitates a more precise comprehension of cost patterns. If missing values are not resolved, statistical analyses may become biased and inaccurate, which could result in findings that are unreliable. We mitigate the effect of data gaps by substituting the mean for missing values, allowing for a more thorough evaluation of average costs. This method preserves the general structure of the information while offering a representative estimate that improves the accuracy of insights into the distribution and patterns of medical costs. (Kumar, 2023)

|  |
| --- |
| **CODE**  # To find out if there are null values  df.isnull()  **Output:**  A table with numbers and letters  Description automatically generated |

|  |
| --- |
| # To find out how many null values does each individual column has  df.isnull().sum()  **Output:**  PersonID 0  age 123  sex 0  bmi 0  children 0  smoker 0  region 0  charges 0  dtype: int64  Based on the output above, I will be able to identify that there are 123 missing values under “age” column  # To find out how many null values does each individual column has  df.isnull().sum()  **Output:**  PersonID 0  age 123  sex 0  bmi 0  children 0  smoker 0  region 0  charges 0  dtype: int64  Based on the output above, I will be able to identify that there are 123 missing values under “age” column |

|  |
| --- |
| # Filling Null Values with the mean of a column  df1 = df.fillna(value=df["age"].mean().astype(int))  df1  **Output:**  A screenshot of a table  Description automatically generated  # To find out how many null values does each individual column in df1 has  df1.isnull().sum()  **Output:**  PersonID 0  age 0  sex 0  bmi 0  children 0  smoker 0  region 0  charges 0  dtype: int64 |

Pre-Processing Task 2 – Organize input in ‘sex’ column

For Python to identify unique values & to standardize them, I will use

* pd.value\_counts(df1['sex'].values) to identify the unique values in the ‘sex column’
* df1['sex'] = df['sex'].map({'female': 'female', 'F': 'female', 'male': 'male', 'M': 'male'}) to replace the values “F” and “M” with “female” & “male” respectively

Detecting unique values assists in deciphering and cleaning the data, while standardizing values ensures that the data is formatted correctly for modelling. Categorical variables in machine learning frequently require encoding in order to train models. Standardized features enable many methods for optimization used in machine learning models to converge quicker and more consistently. By giving the algorithm an easier setting to navigate, it facilitates the optimization process. (Simplilearn, 2023)

|  |
| --- |
| **CODE**  # Count the number of times each unique value appears in "sex" Column  pd.value\_counts(df1['sex'].values)  **Output:**  male 676  female 658  F 4  M 2  dtype: int64 |

|  |
| --- |
| # To replace "F" with "female" and "M" with "male" in "sex" column  df1['sex'] = df['sex'].map({'female': 'female', 'F': 'female', 'male': 'male', 'M': 'male'})  df1['sex']  **Output:**  0 female  1 male  2 male  3 male  4 male  ...  1335 male  1336 female  1337 female  1338 female  1339 female  Name: sex, Length: 1340, dtype: object  # Ensure that “F” and “M” has been amended to “female” & “male” in "sex" Column  pd.value\_counts(df1['sex'].values)  **Output:**  male 678  female 662  dtype: int64 |

Pre-Processing Task 3 – To check for duplicated person ID & removing them

For Python to identify duplicated Person ID & to remove them, I will use

* df1['PersonID\_duplicated'] = df1['PersonID'].duplicated() to check for duplicate “Person ID”
* df1.drop\_duplicates(subset=['PersonID'], keep='first', inplace=True) to remove duplicated ID from the dataset

Removing duplicated IDs is crucial for maintaining data integrity and ensuring the accuracy of analyses. Duplicates can introduce errors and inconsistencies in the dataset, leading to skewed statistical results and potentially biased models. Cleaning the data by eliminating duplicates is fundamental for reliable insights and meaningful exploration of the distribution across unique individuals, contributing to more accurate and actionable conclusions. (Kuan, 2022)

|  |
| --- |
| **CODE**  # To check for duplicate "Person ID"  df1['PersonID\_duplicated'] = df1['PersonID'].duplicated()  # Display rows with duplicated 'PersonID'  duplicated\_rows = df1[df1['PersonID\_duplicated'] == True]  print(duplicated\_rows) |

|  |
| --- |
| # Remove duplicated ID from the dataset  df1.drop\_duplicates(subset=['PersonID'],keep='first',inplace=True)  df1  **Output:**  A screenshot of a graph  Description automatically generated  # Remove "PersonID\_duplicated" column  df1.drop(columns=['PersonID\_duplicated'], inplace=True)  df1  **Output:**  A screenshot of a table  Description automatically generated |

Total wordcount (excluding codes): **308 words**

**Question 2**

**Chart 1:** Scatter plot illustrating the relationship between Medical Cost and Age

A graph showing the number of patients with the number of patients with the number of patients with the number of patients with the number of patients with the number of patients with the number of patients with the

Description automatically generated

The scatter plot illustrates a clear correlation between age and medical costs. As the age increases, there is a noticeable upward trend in medical charges. This is consistent with the generally believed concept that health problems are more common in older adults, which raises the need for medical care and, in turn, the expense of that treatment. (Chen et al., 2023)

Furthermore, the scatter plot distinguishes smokers from non-smokers with its distinct red and blue dots. From the chart, it is obvious that smokers (distinguish by red dots) exhibit a clear tendency to incur higher medical charges compared to non-smokers (distinguish by blue dots). This visual contrast strongly demonstrates that smoking is a major contributing cause to higher medical expenses.

The vertical stretch of red and blue dots around age 40, attributed to the replacement of missing values with the mean of the "age" column. This adjustment shows the accurate representation of relationship between smoking status and medical costs but not for the age and medical costs.

While interpreting the plot, it is crucial to recognize the presence of outliers, particularly the seven smokers incurring medical costs exceeding $50,000. These outliers warrant closer examination as they may be undergoing intensive and costly treatments for severe smoking-related conditions. Investigating these cases can provide valuable insights into the specific health challenges associated with smoking and inform strategies for preventive healthcare. (Izumi et al., 2001)

|  |
| --- |
| **CODE**  import matplotlib.pyplot as plt  from matplotlib.ticker import FuncFormatter  import seaborn as sns  plt.figure(figsize=(10, 6))  sns.set(style="whitegrid")  # Define the order of hue levels  hue\_order = ['Smoker', 'Non-Smoker']  # Create Scatter plot  plt.figure(figsize=(10, 6))  scatter\_plot = sns.scatterplot(x='age', y='charges', hue='smoker', data=df1, palette={'yes': 'red', 'no': 'blue'})  # Function to format y-ticks with a thousand separator  def format\_thousands(value, pos):  return '{:,.0f}'.format(value)  # Apply the formatter to the y-axis  scatter\_plot.yaxis.set\_major\_formatter(FuncFormatter(format\_thousands))  # Add labels and Title  plt.title('Medical Charges according to age', fontsize=18)  plt.xlabel('Age', fontsize=12)  plt.ylabel('Charges', fontsize=12) |

|  |
| --- |
| # Manually create legend with desired colours  legend\_labels = ['Smoker', 'Non-Smoker']  legend\_colors = {'Smoker': 'red', 'Non-Smoker': 'blue'}  legend\_handles = [plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=legend\_colors[label], markersize=10)  for label in legend\_labels]  # Customize legend  plt.legend(legend\_handles, legend\_labels, title='Smoking Status', loc='upper right', bbox\_to\_anchor=(1.25, 1))  # Show the plot  plt.show() |

**Chart 2:** Bar Chart illustrating the relationship between Region, Gender and Medical Cost

A graph of different colored bars

Description automatically generated

The analysis of regional and gender-based variations in medical costs reveals compelling insights. Notably, the average medical expenses in the Northeast and Southeast regions surpass those in other areas. In the Northeast, states like New York, characterized by elevated living costs and cutting-edge healthcare facilities, contribute to the overall higher medical expenditure. Additionally, the Southeast, marked by a warmer climate, sees an increased prevalence of heat-related illnesses during summers. This climatic factor could be a significant contributor to the augmented medical costs observed in the region. (Rodriguez, 2023)

The gender disparity in medical costs, with males incurring higher expenses, aligns with established patterns. This phenomenon may be attributed to the tendency of men to partake in riskier behaviors, leading to a higher likelihood of accidents and injuries. Such behavioral tendencies contribute to the increased healthcare utilization by males. (Udry, 1998).

|  |
| --- |
| **CODE**  # Calculate average costs by region  average\_costs\_gender = df1.groupby(['region', 'sex'])['charges'].mean().reset\_index()  # Print the average costs by region  print(average\_costs\_gender)  **Output:**  region sex charges  0 northeast female 12953.203151  1 northeast male 13854.005374  2 northwest female 12479.870397  3 northwest male 12354.119575  4 southeast female 13499.669243  5 southeast male 15879.617173  6 southwest female 11274.411264  7 southwest male 13412.883576 |

|  |
| --- |
| import seaborn as sns  import matplotlib.pyplot as plt  # Calculate average costs by region and gender  average\_costs\_gender = df1.groupby(['region', 'sex'])['charges'].mean().reset\_index()  # Create a bar plot  plt.figure(figsize=(12, 8))  ax = sns.barplot(x='region', y='charges', hue='sex', data=average\_costs\_gender, palette={'female': 'pink', 'male': 'lightgreen'})  # Add labels and title  plt.xlabel('Region', fontsize=12)  plt.ylabel('Average Medical Cost', fontsize=12)  plt.title('Average Medical Cost by Region and Gender', fontsize=18)  # Add average cost annotations  for p in ax.patches:  height = p.get\_height()  ax.annotate(f'${height:.2f}', (p.get\_x() + p.get\_width() / 2., height),  ha='center', va='bottom', fontsize=12, color='black')  # Show the plot  plt.show() |

**Chart 3:** Histogram illustrating the relationship between number of children and average Medical Cost

A graph showing the average medical cost by number of children

Description automatically generated

The depicted chart illustrates a noticeable upward trend in average medical costs as the number of children in a family increase from 0 to 5. This trend culminates in the highest average cost at 3 children, reaching $15,355.32. Interestingly, there is a clear oddity that shows a large decline in average medical costs when there are five children. This anomaly should be interpreted with caution because it could be the result of a small sample size or unique conditions impacting the small number of families in the dataset that have exactly five children. It's important to take into account the likelihood that families with more children may have different healthcare routines, which could result in different cost trends. Additionally, further investigation into the demographic and socioeconomic characteristics of these families would enhance the accuracy of any conclusions drawn from the data.

Total wordcount (excluding codes): **513 words**

|  |
| --- |
| **CODE**  import seaborn as sns  import matplotlib.pyplot as plt  # Calculate average medical costs by number of children  average\_costs = df1.groupby('children')['charges'].mean().reset\_index()  # Print the average medical costs by number of children  print(average\_costs)  **Output:**  children charges  0 0 12377.971169  1 1 12731.171832  2 2 15073.563734  3 3 15355.318367  4 4 13850.656311  5 5 8786.035247  # Create a histogram  plt.figure(figsize=(10, 6))  ax = sns.barplot(x='children', y='charges', data=average\_costs, palette='viridis', edgecolor='black')  # Add labels above each bar  for index, row in average\_costs.iterrows():  ax.text(index, row['charges'] + 200, f'${row["charges"]:.2f}', ha='center', va='bottom', color='black', fontsize=10)  # Labels and title  plt.xlabel('Number of Children', fontsize=12)  plt.ylabel('Average Medical Cost', fontsize=12)  plt.title('Average Medical Cost by Number of Children', fontsize=18)  # Show the plot  plt.show() |

**Question 3**

The dataset, df1, undergoes a preprocessing phase where features (X) and the target variable (y) are segregated, with the latter being "smoker," our prediction target. Notably, categorical attributes like "sex" and "region" are earmarked for encoding. Employing a ColumnTransformer, the dataset is systematically preprocessed, applying one-hot encoding exclusively to categorical features while leaving others unaltered. The subsequent step involves the instantiation of a Decision Tree Classifier, with a predefined random state for result reproducibility.

To streamline the workflow, a scikit-learn Pipeline is instituted, seamlessly integrating data preprocessing and model training. The dataset is judiciously split into training and testing subsets using train\_test\_split, laying the foundation for robust model evaluation. Subsequently, the pipeline is trained on the designated training data, encapsulating both preprocessing and the Decision Tree model.

Predictions are then generated on the test set using the trained pipeline, allowing for a comprehensive assessment of model performance. The Decision Tree's intricacies are brought to light through visualization, orchestrated by the plot\_tree function from scikit-learn. This visualization is meticulously refined for enhanced interpretability, encompassing adjustments like setting class names, node filling, and corner rounding.

This approach offers a comprehensive view of the Decision Tree model, including its structure and feature importance. The saved image can be inspected to understand the decision-making process of the model and identify key features influencing the prediction of "smoker" status.

Wordcount: **224 words**

|  |
| --- |
| **CODE**  from sklearn.model\_selection import train\_test\_split  from sklearn.tree import DecisionTreeClassifier, plot\_tree  from sklearn.preprocessing import OneHotEncoder  from sklearn.compose import ColumnTransformer  from sklearn.pipeline import Pipeline  import pandas as pd  import matplotlib.pyplot as plt  # Replace NaN values with the mean of each column  df1 = df.fillna(value=df["age"].mean().astype(int))  # Replace "F" with "female" and "M" with "male" in "sex" column  df1['sex'] = df1['sex'].map({'female': 'female', 'F': 'female', 'male': 'male', 'M': 'male'})  # Remove duplicated ID from the dataset  df1.drop\_duplicates(subset=['PersonID'], keep='first', inplace=True)  # Separate features (X) and target variable (y)  X = df1.drop('smoker', axis=1)  y = df1['smoker']  # Specify categorical features for encoding  categorical\_features = ['sex', 'region'] |

|  |
| --- |
| # Create a column transformer for preprocessing  preprocessor = ColumnTransformer(  transformers=[  ('cat', OneHotEncoder(drop='first'), categorical\_features)  ],  remainder='passthrough'  )  # Preprocess the data and extract feature names  X\_preprocessed = preprocessor.fit\_transform(X)  feature\_names = preprocessor.get\_feature\_names\_out(X.columns)  # Split the data into training and testing sets  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_preprocessed, y, test\_size=0.2, random\_state=42)  # Create a decision tree classifier  dt\_model = DecisionTreeClassifier(random\_state=42)  # Fit the model on the training data  dt\_model.fit(X\_train, y\_train)  # Print a message to indicate that the model has been trained  print("Decision Tree Model trained successfully.")  # Generate predictions on the test set  y\_pred = dt\_model.predict(X\_test)  # Print a message to indicate that predictions have been generated  print("Predictions generated successfully.") |
| # Get feature names after one-hot encoding  encoded\_feature\_names = preprocessor.named\_transformers\_['cat'].get\_feature\_names\_out(categorical\_features)  feature\_names = list(encoded\_feature\_names) + list(X.columns[len(categorical\_features):])  # Plot the decision tree  plt.figure(figsize=(20,10))  plot\_tree(dt\_model, feature\_names=feature\_names, class\_names=["non-smoker", "smoker"], filled=True, rounded=True)  plt.savefig('decision\_tree.png', dpi=500)  plt.show() |

A diagram of a network

Description automatically generatedo**Question 4**

|  |
| --- |
| # Combine importances for sex\_male and sex  feature\_importance\_df.loc[feature\_importance\_df['Feature'].str.startswith('sex'), 'Feature'] = 'sex'  # Combine importances for region\_northwest, region\_southeast, and region\_southwest  feature\_importance\_df.loc[feature\_importance\_df['Feature'].str.startswith('region'), 'Feature'] = 'region'  # Sum up the importances for the combined categories  feature\_importance\_df = feature\_importance\_df.groupby('Feature', as\_index=False)['Importance'].sum()  # Print or visualize the updated feature importances  print(feature\_importance\_df)  **Output:**  Feature Importance  0 bmi 0.039370  1 charges 0.760471  2 children 0.159466  3 region 0.005250  4 sex 0.035443 |

The decision tree prominently reveals the distinction between smokers, represented by blue leaf nodes, and non-smokers, represented by orange leaf nodes. The root node hinges on the condition "charges <= 15254.557," making it clear that medical charges play a pivotal role in determining an individual's smoking status. The model assigns a substantial importance score of 0.76 to the "charges" feature, underscoring its significance in the classification process. Higher medical charges are indicative of a likely association with smokers, aligning with the well-established link between smoking and health-related costs.

The second decision node refines the classification further based on charges. Specifically, if charges are less than or equal to 14453.74, the model incorporates additional features—Children and BMI. This suggests that while charges remain pivotal, the number of children and BMI contribute crucial information to the decision-making process.

Children emerge as the second most crucial feature, with an importance score of 0.159. This underscores that the number of children an individual has plays in predicting smoking status. The model recognizes the impact of family dynamics, highlighting the correlation between the presence of children and smoking behaviour.

While BMI, Sex, and Region maintain their relevance in predicting smoking status, their importance scores are comparatively lower than charges and the number of children.

Total Wordcount: **210 words**

**Question 5**

Decision trees prove to be invaluable tools for exploratory data analysis (EDA), extending beyond their traditional predictive role. Their feature importance ranking, easily derived from the tree structure, provides a clear understanding of factors influencing the target variable, enhancing interpretability (Decision Tree Vs Logistic Regression Feature Importances, n.d.).

The branching nature of decision trees unveils intricate patterns and connections within the data, offering a visual representation that comprehensively illustrates how various features interact. This transparency aids in identifying relationships and contributions, making decision trees indispensable for data exploration (Dawood, 2023).

Unlike linear models, decision trees adeptly capture non-linear relationships between variables, revealing nuanced dependencies that simpler analytical approaches might overlook. This is particularly beneficial in EDA, where complex relationships are common (Parihar, 2021).

Decision trees excel in outlier detection through their creation of distinct decision routes. By scrutinizing these routes, unusual patterns or deviations from the norm are easily identified, transforming decision trees into effective diagnostic tools for anomaly detection (Are Decision Trees Robust to Outliers, n.d.).

The representation of variable interactions in decision nodes adds depth to the analysis, offering nuanced insights into conditional dependencies within the data. This enriches the overall understanding of relationships between different features, contributing to a holistic EDA (Sen, 2023).

Decision trees' natural handling of categorical variables eliminates the need for one-hot encoding, streamlining the analysis process. Their adaptability and efficiency make them well-suited for datasets with a mix of numerical and categorical variables (Kinasih, 2023).

Graphical representation enhances the communication of complex relationships within the data. Decision trees' visual appeal simplifies the conveyance of ideas, making analytical findings more accessible to stakeholders less familiar with advanced statistical approaches (Thorn, 2021).

In conclusion, decision trees emerge as versatile instruments in EDA, offering transparency, non-linear insights, outlier detection, and adaptability, guiding analysts through the intricate landscapes of diverse datasets.

Total Wordcount: **304 words**

**Reference(s):**

Anello, E. (n.d.). 7 steps to mastering data cleaning and preprocessing techniques - KDNuggets. KDnuggets.

<https://www.kdnuggets.com/2023/08/7-steps-mastering-data-cleaning-preprocessing-techniques.html>

Are decision trees robust to outliers. (n.d.). Data Science Stack Exchange. <https://datascience.stackexchange.com/questions/37394/are-decision-trees-robust-to-outliers>

Chen, J., Zhao, M., Zhou, R., Ou, W., & Yao, P. M. (2023). How heavy is the medical expense burden among the older adults and what are the contributing factors? A literature review and problem-based analysis. Frontiers in Public Health, 11. <https://doi.org/10.3389/fpubh.2023.1165381>

Dawood, M. (2023, May 30). Decision Tree: Building Machine learning model. <https://www.linkedin.com/pulse/decision-tree-building-machine-learning-model-muhammad-dawood>

Decision tree vs logistic regression feature importances. (2023, March). Data Science Stack Exchange.

<https://datascience.stackexchange.com/questions/116549/decision-tree-vs-logistic-regression-feature-importances#:~:text=In%20decision%20trees%2C%20feature%20importance,splitting%20on%20a%20particular%20feature>.

Izumi, Y., Tsuji, I., Ohkubo, T., Kuwahara, A., Nishino, Y., & Hisamichi, S. (2001). Impact of smoking habit on medical care use and its costs: a prospective observation of National Health Insurance beneficiaries in Japan. International Journal of Epidemiology, 30(3), 616–621. [https://doi.org/10.1093/ije/30.3.616](https://doi.org/10.1093/ije/30.3.616%20)

Kinasih, D. a. S. (2023, August 27). Categorical feature in Decision Tree Classifier - Dyah Ayu Sekar Kinasih - medium. Medium.

<https://medium.com/@dyahayusekarkinasih/categorical-feature-in-decision-tree-classifier-3ad0c42c6dcc#:~:text=Yes%2C%20decision%20trees%20are%20capable,training%20or%20testing%20the%20model>.

Kuan, F. (2022, April 20). Data Cleaning - Remove duplicates. Mage. <https://www.mage.ai/blog/data-cleaning-remove-duplicates>

Kumar, A. (2023, March 26). Python - Replace Missing Values with Mean, Median & Mode - Analytics Yogi. Analytics Yogi.

<https://vitalflux.com/pandas-impute-missing-values-mean-median-mode/#:~:text=Mean%20imputation%20is%20often%20used,to%20outliers%20than%20the%20mean>.

Parihar, G. (2021, December 14). Machine Learning: decision tree classification - Analytics Vidhya - medium. Medium. <https://medium.com/analytics-vidhya/machine-learning-decision-tree-classification-1a248e95bc03#:~:text=Decision%20trees%20are%20easy%20to,It%20can%20overfit%20noisy%20data>.

Python Machine Learning Decision Tree. (n.d.). <https://www.w3schools.com/python/python_ml_decision_tree.asp>

Rodriguez, C. (2023, June 27). The health care costs of extreme heat. Center for American Progress.

<https://americanprogress.org/article/the-health-care-costs-of-extreme-heat/>

Sen, V. (2023, October 5). “NO” need to check for multicollinearity or remove correlated variables explicitly when using decision trees. <https://www.linkedin.com/pulse/need-check-multicollinearity-remove-correlated-variables-sen#:~:text=Decision%20trees%20are%20inherently%20capable,implicitly%20during%20the%20training%20process>.

Simplilearn. (2023, February 21). Data standardization: how it’s done & why it’s important. Simplilearn.com.

<https://www.simplilearn.com/what-is-data-standardization-article#:~:text=Data%20standardization%20converts%20data%20into,to%20communicate%20and%20exchange%20information>.

Thorn, J. (2021, December 13). Decision Trees explained - towards data science. Medium. <https://towardsdatascience.com/decision-trees-explained-3ec41632ceb6#:~:text=The%20main%20advantage%20of%20decision,require%20less%20data%20to%20train>.

Udry, J. R. (1998). Why are males injured more than females? Injury Prevention, 4(2), 94–95. <https://doi.org/10.1136/ip.4.2.94>