Teach me this code as if you have oral exam on this : # Assignment No. 1 Data Wrangling I: # Program by Prof. Rahul Dagade & Prof. Varsha Nale #import libraries import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Read Dataset df=pd.read_csv(r"placement_data.csv") print("Placement dataset is successfully loaded into Data Frame....") #Display information of dataset print("Information of Dataset: \n", df.info) print("Shape of Dataset (row x column):",df.shape) print("Columns Name:", df.columns) print("Total elements in dataset:", df.size) print("Datatype of attributes (columns):", df.dtypes) print("First 5 rows: \n", df.head().T) print("Last 5 rows:\n",df.tail().T) print("Any 5 rows: \n",df.sample(5).T) #Display statistical information of dataset print("Statistical information of Numerical Columns: \n", df.describe()) #Display null values print("Total Number of Null Values in Dataset:", df.isna().sum()) # Data type conversion print("Converting Data type of variables: ") df["sl_no"]=df["sl_no"].astype("int8") print("Check Datatype of sl_no",df.dtypes) df["ssc_p"]=df["ssc_p"].astype("int8") print("Check Datatype of ssc_p",df.dtypes) # Label Encoding Conversion of Categorical to Quantitative print("Encoding using Label Encoding (Cat Codes)") df["gender"] =df["gender"].astype("category") print("Data types of Gender=", df.dtypes["gender"]) df["gender"]=df["gender"].cat.codes print("Data types of gender after label encoding =", df. dtypes['gender']) print("Gender Values:",df["gender"].unique()) # Normalization print("Normalization using Min-Max Feature Scaling: ") df["salary"]=(df["salary"]-df["salary"].min())/(df ["salary"].max()-df["salary"].min()) print(df.head().T)

ChatGPT

2 / 2

Sure! Let's go through the code step by step.

First, we need to import the required libraries. In this code, we are using numpy, pandas, seaborn, and matplotlib.pyplot. These libraries provide various functionalities for data manipulation, analysis, and visualization.

pythonCopy code

```
import as import as import as import as
```

Next, we read the dataset from a CSV file called "placement_data.csv" using pandas' `read_csv()` function and store it in a DataFrame called `df`.

pythonCopy code

```
r"placement_data.csv" print "Placement dataset is successfully loaded into Data
Frame...."
```

To understand the dataset, we display some basic information about it using various pandas functions.

pythonCopy code

```
print "Information of Dataset: \n" print "Shape of Dataset (row x column):"
print "Columns Name:" print "Total elements in dataset:" print "Datatype of
attributes (columns):" print "First 5 rows: \n" print "Last 5 rows:\n"
print "Any 5 rows: \n" 5
```

The `info()` function provides a summary of the dataset, including the number of rows, columns, column names, and the data type of each column.

The `shape` attribute gives the dimensions of the dataset (number of rows x number of columns).

The `columns` attribute lists the names of all the columns in the dataset.

The `size` attribute gives the total number of elements in the dataset.

The `dtypes` attribute shows the data type of each column in the dataset.

The `head()` function displays the first 5 rows of the dataset.

The `tail()` function displays the last 5 rows of the dataset.

The `sample()` function displays any 5 random rows from the dataset.

After displaying the basic information, we move on to analyze the statistical information of the numerical columns in the dataset using the `describe()` function.

pythonCopy code

```
print "Statistical information of Numerical Columns: \n"
```

The `describe()` function provides various statistical measures like count, mean, standard deviation, minimum value, 25th percentile, median, 75th percentile, and maximum value for the numerical columns in the dataset.

Next, we check for null values in the dataset using the `isna().sum()` function.

pythonCopy code

```
print "Total Number of Null Values in Dataset:" sum
```

The `isna()` function returns a DataFrame with the same shape as the original dataset, where each element is a Boolean value indicating whether the corresponding element in the original dataset is missing or not. The `sum()` function then sums up the number of missing values in each column.

After that, we perform data type conversions on specific columns in the dataset.

pythonCopy code

```
print "Converting Data type of variables: " "sl_no" "sl_no" "int8" print "Check Datatype
of sl_no" "ssc_p" "ssc_p" "int8" print "Check Datatype of ssc_p"
```

Here, we convert the "sl_no" column and "ssc_p" column to the int8 data type using the `astype()` function. This conversion helps in reducing memory usage if the values in these columns can be represented within the int8 range.

Next, we perform label encoding on a categorical column to convert it into a quantitative representation.

pythonCopy code

```
print "Encoding using Label Encoding (Cat Codes)" "gender" "gender" "category"
print "Data types of Gender=" "gender" "gender" "gender" print "Data
types of gender after label encoding =" 'gender' print "Gender Values:"
 "gender"
```

In this code snippet, we first convert the "gender" column to the category data type. Then, we use the `cat.codes` attribute to assign a numerical code to each unique category in the "gender" column. This effectively converts the categorical values into numeric representations. Finally, we display the unique values of the "gender" column after label encoding.

Lastly, we perform normalization on the "salary" column using the Min-Max feature scaling technique.

pythonCopy code

```
print "Normalization using Min-Max Feature Scaling: " "salary" "salary" "salary" min
 "salary" max "salary" min print
```

Here, we subtract the minimum value of the "salary" column from each element and divide it by the range (maximum value minus minimum value) of the "salary" column. This scales the values in the "salary" column between 0 and 1, making them normalized.

That's the end of the code explanation!

MY

```
#Assignment No. 2 Data Wrangling II Program by Prof. Rahul Dagade & Prof. Varsha Nale def
RemoveOutlier(df,var): Q1 =df[var].quantile(0.25) Q3 =df[var].quantile(0.75) IQR = Q3-Q1 high,
low=Q3+1.5*IQR, Q1-1.5*IQR df=df[((df[var] >= low) & (df[var] <= high))] print("Outliers removed
in", var) return df def DisplayOutliers(df, message): fig, axes = plt.subplots(2,2) fig.suptitle(message)
sns.boxplot(data=df, x="raisedhands", ax=axes[0,0]) sns.boxplot(data=df, x="VisITedResources",
ax=axes[0,1]) sns.boxplot(data=df, x="AnnouncementsView", ax=axes[1,0]) sns.boxplot(data=df,
x="Discussion", ax=axes[1,1]) fig.tight_layout() plt.show() #import libraries import numpy as np
import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Read Dataset
df=pd.read_csv(r"C:\Users\niraj\Downloads\student_data.csv") print("Student Academic Performance
Dataset is successfully loaded...") #Display information of dataset print("Information of Dataset\:n",
df.info) print("Shape of Dataset (row x column):", df.shape) print("Columns Name:", df.columns)
print("Total elements in dataset:", df.size) print("Datatype of attributes {columns}:", df.dtypes)
print("First 5 rows: \n",df.head().T) print("Last 5 rows:\n",df.tail().T) print("Any 5 rows:
\n",df.sample(5).T) #Display statistical information of dataset print("Statistical information of
```

Numerical Columns: \n",df.describe()) #Display null values print("Total Number of Null values in Dataset:\n", df.isna().sum()) #Handling outliers DisplayOutliers(df, "Before removing Outliers") df=RemoveOutlier(df, "raisedhands") df=RemoveOutlier(df, "VisITedResources") df=RemoveOutlier(df, "AnnouncementsView") df=RemoveOutlier(df, "Discussion") DisplayOutliers(df, "After removing Outliers") #Conversion of Categorical to Quantitative (Encoding) df["gender"]=df["gender"].astype("category") df["gender"]=df["gender"].cat.codes print("Data types of gender after label encoding =",df.dtypes['gender']) print("Gender Values:", df["gender"].unique()) sns.boxplot(data=df, x="gender", y="raisedhands", hue="gender") plt.title("Boxplot with 2 variables gender and raisedhands") plt.show() sns.boxplot(data=df,x="NationalITy", y="Discussion", hue="gender") plt.title("Boxplot with 3 variables gender, nationality, discussion") plt.show() print ("Relationship between variables using Scatterplot:") sns.scatterplot(data=df,x= "raisedhands", y="VisITedResources") plt.title("Scatterplot for raisedhands, VisITedResources") plt.show()

ChatGPT

Sure! Let's break down the code step by step:

First, we have two functions defined: `RemoveOutlier(df, var)` and `DisplayOutliers(df, message)`. These functions are used for outlier removal and displaying outliers respectively.

pythonCopy code

```
def RemoveOutlier 0.25 0.75
1.5 1.5 print "Outliers removed
in" return def DisplayOutliers 2 2
"raisedhands" 0 0 "VisITedResources"
0 1 "AnnouncementsView" 1 0
"Discussion" 1 1
```

The `RemoveOutlier()` function takes two arguments: `df` (the DataFrame) and `var` (the variable/column name). It removes outliers from the specified column using the Interquartile Range (IQR) method. It calculates the first quartile (Q1), third quartile (Q3), and IQR, and then defines the upper and lower thresholds for outliers. It filters the DataFrame based on these thresholds and returns the modified DataFrame.

The `DisplayOutliers()` function takes two arguments: `df` (the DataFrame) and `message` (a string for the plot title). It displays boxplots for four columns ("raisedhands", "VisITedResources", "AnnouncementsView", and "Discussion") using seaborn's `boxplot()` function. The boxplots are arranged in a 2x2 grid using subplots. The `message` string is used as the main title of the figure. The function then shows the plot.

Next, we import the required libraries: numpy, pandas, seaborn, and matplotlib.pyplot.

pythonCopy code

```
import as import as import as import as
```

We read the dataset from a CSV file called "student_data.csv" using pandas' `read_csv()` function and store it in a DataFrame called `df`.

pythonCopy code

```
r"C:\Users\niraj\Downloads\student_data.csv" print "Student Academic Performance
Dataset is successfully loaded..."
```

To understand the dataset, we display some basic information about it using various pandas functions.

pythonCopy code

```
print "Information of Dataset:\n" print "Shape of Dataset (row x column):"
print "Columns Name:" print "Total elements in dataset:" print "Datatype of
attributes {columns}:" print "First 5 rows:\n" print "Last 5 rows:\n"
print "Any 5 rows:\n" 5
```

These functions provide information about the dataset, such as the number of rows and columns, column names, data types, and a preview of the data.

We then display the statistical information of the numerical columns using the `describe()` function.

pythonCopy code

```
print "Statistical information of Numerical Columns:\n"
```

The `describe()` function provides various statistical measures like count, mean, standard deviation, minimum value, 25th percentile, median, 75th percentile, and maximum value for the numerical columns in the dataset.

Next, we check for null values in the dataset using the `isna().sum()` function.

pythonCopy code

```
print "Total Number of Null values in Dataset:\n" sum
```

The `isna()` function returns a DataFrame with the same shape as the original dataset, where each element is a Boolean value indicating whether the corresponding element in the original dataset is missing or not. The `sum()` function then sums up the number of missing values in each column.

After that, we handle outliers in the dataset using the `DisplayOutliers()` and `RemoveOutlier()` functions.

pythonCopy code

```
"Before removing Outliers" "raisedhands"
"VisITedResources" "AnnouncementsView"
"Discussion" "After removing Outliers"
```

We display the outliers before and after removing them using boxplots. The `DisplayOutliers()` function is called with the original `df` DataFrame and a message for the plot title. Then, the `RemoveOutlier()` function is called for each column to remove the outliers from the DataFrame. The modified DataFrame is assigned back to `df`.

Next, we perform label encoding on the "gender" column to convert it into a quantitative representation.

pythonCopy code

```
"gender" "gender" "category" "gender" "gender" print "Data types of
gender after label encoding =" 'gender' print "Gender Values:" "gender"
```

Here, we convert the "gender" column to the category data type and then use the `cat.codes` attribute to assign a numerical code to each unique category. We display the data type of the "gender" column after label encoding and the unique values of the "gender" column.

Finally, we use seaborn to create boxplots and scatterplots to visualize relationships between variables.

pythonCopy code

```
"gender" "raisedhands" "gender" "Boxplot with 2 variables
gender and raisedhands" "NationalITy" "Discussion" "gender"
"Boxplot with 3 variables gender, nationality, discussion" print "Relationship between
variables using Scatterplot:" "raisedhands" "VisITedResources"
"Scatterplot for raisedhands, VisITedResources"
```

These plots show the relationship between variables. The first boxplot shows the relationship between "gender" and "raisedhands". The second boxplot shows the relationship between "gender", "NationalITy", and "Discussion". Finally, the scatterplot shows the relationship between "raisedhands" and "VisITedResources".

That's the end of the code explanation!

MY

#Assignment number: 3 Statistical Information #Program by Prof. Rahul Dagade & Prof. Varsha Nale #import libraries import pandas as pd import seaborn as sns import matplotlib.pyplot as plt import numpy as np #Read dataset df = pd.read_csv(r"C:\Users\niraj\Downloads\Employee_Salary.csv") print("Employee_Salary Dataset is successfully loaded.\n") #Display information of dataset print("Iris Dataset is successfully loaded....") print("Information of Dataset:\n", df.info) print("Shape of Dataset (row x column):", df.shape) print("Columns Name: df.columns") print("Total elements in dataset:", df. size) print("Datatype of attributes (columns):", df.dtypes) print("First 5 rows: \n", df.head().T) print("Last 5 rows: \n", df.tail().T) print("Any 5 rows: \n", df.sample(5).T) print("Statistical information of Numerical Columns: \n",) columns = ["Experience_Years", "Age", "Salary"] print("{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}".format("Columns", "Min","Max","Mean","Median","STD")) for column in columns: m1,m2,m3=df[column].min(),df[column].max(),df [column].mean() m4,s=df[column].median(),df[column].std() print("{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}".format(column, m1, m2, m3, m4,5)) print("roupwise

```
Statistical Summary...") columns=["Experience_Years", "Age", "Salary"] for column in columns:
print("\n----------------------",column,"-------------------\n")
print("{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}".format("Columns","Min","Max","Mean","Median","STD"))
m1=df[column].groupby(df ["Gender"]).min() m2=df[column].groupby(df["Gender"]).max()
m3=df[column].groupby(df["Gender"]).mean() m4=df[column].groupby(df["Gender"]).median()
s=df[column].groupby(df["Gender"]).std()
print("{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}".format("Female",m1[0],m2[0],m3[0],m4[0],s[0]))
print("{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}".format("Male",m1[1],m2[1],m3[1],m4[1],s[1])) X = ["min",
"max","mean", "median", "std"] features = ["Salary", "Age","Experience_Years"] df1 =
pd.DataFrame(columns=["min", "max", "mean", "median", "std"]) for var in features: df1["min"]
=df[var].groupby(df["Gender"]).min() df1["max"] =df[var].groupby(df["Gender"]).max() df1["mean"]
=df[var].groupby(df["Gender"]).mean() df1["median"]=df [var].groupby(df["Gender"]).median ()
df1["std"]=df[var].groupby(df["Gender"]).std() X_axis=np.arange(len(X)) plt.bar (X_axis-0.2,df1.iloc[0],
0.4, label = "Female") plt.bar(X_axis+0.2,df1.iloc[1], 0.4, label = "Male") plt.xticks(X_axis, X)
plt.xlabel("Statistical information") plt.ylabel(var) plt.title("Groupwise Statistical Information of
Employee Salary Dataset") plt.legend() plt.show()
```

ChatGPT

Sure! Let's go through the code step by step:

First, the required libraries are imported: pandas, seaborn, matplotlib.pyplot, and numpy.

pythonCopy code

```python
import  as import  as import  as import  as
```

Next, the dataset is read from a CSV file called "Employee_Salary.csv" using pandas' `read_csv()` function, and it is stored in a DataFrame called `df`.

pythonCopy code

```python
r"C:\Users\niraj\Downloads\Employee_Salary.csv" print "Employee_Salary Dataset is
successfully loaded.\n"
```

Information about the dataset is displayed using various pandas functions.

pythonCopy code

```python
print "Iris Dataset is successfully loaded..." print "Information of Dataset:\n" print "Shape of
Dataset (row x column):" print "Columns Name: df.columns" print "Total elements in
dataset:" print "Datatype of attributes (columns):" print "First 5 rows: \n"
print "Last 5 rows: \n" print "Any 5 rows: \n" 5
```

These functions provide information about the dataset, such as the number of rows and columns, column names, data types, and a preview of the data.

The statistical information of numerical columns is computed and displayed using a for loop.

pythonCopy code

```
print "Statistical information of Numerical Columns: \n"              "Experience_Years" "Age" "Salary"
print "{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}" format "Columns"
"Min" "Max" "Mean" "Median" "STD"  for           in
                          min              max
                                                 print "{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}" format
                  5
```

This loop calculates and prints the minimum, maximum, mean, median, and standard deviation for each column in the `columns` list.

Next, a groupwise statistical summary is computed and displayed.

pythonCopy code

```
print "roupwise Statistical Summary..."          "Experience_Years" "Age" "Salary" for          in
          print "\n----------------------"          "-------------------\n"
print "{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}" format "Columns" "Min" "Max" "Mean" "Median" "STD"
                               "Gender"  min                              "Gender"  max
                               "Gender"                                   "Gender"
                             "Gender"
print "{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}" format "Female"     0      0      0      0    0
print "{:<20}{:<10}{:<10}{:<20}{:<10}{:<20}" format "Male"     1      1      1      1    1
```

This loop calculates and prints the minimum, maximum, mean, median, and standard deviation for each column grouped by the "Gender" column.

Finally, a bar plot is created to visualize the group

MY

#Assignment Number: 4 Boston Housing (Linear Regression)Program by Prof. Rahul Dagade & Prof. Varsha Nale def RemoveOutlier(df, var): Q1=df[var].quantile(0.25) Q3=df[var].quantile(0.75) IQR =Q3-Q1 high, low=Q3+1.5*IQR, Q1-1.5+IQR df=df[((df[var] >= low) & (df[var] <=high))] return df def DisplayOutlier(df, msg): fig,axes=plt.subplots(1,2) fig.suptitle(msg) sns.boxplot(data=df, x="rm", ax=axes[0]) sns.boxplot(data = df, x="lstat", ax=axes[1]) fig.tight_layout() plt.show() #import libraries import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Read Dataset df=pd.read_csv (r"C:\Users\niraj\Downloads\Boston.csv") print("Boston dataset is successfully loaded") #Display information of dataset print("Information of Dataset:\n", df.info) print("Shape of Dataset row x column):", df.shape) print("Columns Name:", df.columns) print("Total elements in dataset:",df.size) print("Datatype of attributes (columns):" ,df.dtypes) print("First 5 rows: \n", df.head().T) print("last 5 rows:\n",df.tail().T) print("Any 5 rows: \n",df.sample(5).T) #Find missing values print ("Missing values") print(df.isnull().sum()) #Find correlation matrix- print("Finding correlation matrix using heatmap:") sns.heatmap(df.corr(),annot=True) plt.show() #Finding and removing outliers print("Finding and removing outliers: ") DisplayOutlier(df, "Before removing

Outliers:") print("Identifying overall outliers in Column Name variables") df = RemoveOutlier(df, "lstat") df = RemoveOutlier(df,"rm") DisplayOutlier(df,"After removing Outliers") #Split the data into inputs and outputs x=df[["rm","lstat"]] #input data y=df["medv"] #output data #Training and testing data from sklearn.model_selection import train_test_split #Assign test data size 20% x_train, x_test, y_train, y_test =train_test_split(x,y,test_size=0.20, random_state=0) #Apply linear regression model on training data from sklearn.linear_model import LinearRegression model=LinearRegression().fit(x_train, y_train) y_pred=model.predict(x_test) #Display accuracy of the model from sklearn.metrics import mean_absolute_error print("MAE:",mean_absolute_error(y_test,y_pred)) print("Model Score:",model.score(x_test,y_test)) #Test the model using user input print("Predict House Price by giving User Input:") features= np.array([[6,19]]) prediction= model.predict(features) print("Prediction: ()".format(prediction))

ChatGPT

Sure! Let's go through the code step by step:

First, the required libraries are imported: numpy, pandas, seaborn, and matplotlib.pyplot.

pythonCopy code

```
import         as   import         as   import          as   import                    as
```

Next, the dataset is read from a CSV file called "Boston.csv" using pandas' `read_csv()` function, and it is stored in a DataFrame called `df`.

pythonCopy code

```
                  r"C:\Users\niraj\Downloads\Boston.csv"  print  "Boston dataset is successfully loaded"
```

Information about the dataset is displayed using various pandas functions.

pythonCopy code

```
print  "Information of Dataset:\n"          print  "Shape of Dataset (row x column):"
print  "Columns Name:"               print  "Total elements in dataset:"          print  "Datatype of
attributes (columns):"            print  "First 5 rows: \n"              print  "last 5 rows:\n"
print  "Any 5 rows: \n"             5
```

These functions provide information about the dataset, such as the number of rows and columns, column names, data types, and a preview of the data.

The missing values in the dataset are checked using the `isnull().sum()` function.

pythonCopy code

```
print  "Missing values"  print             sum
```

The correlation matrix is computed and displayed using a heatmap from seaborn.

pythonCopy code

```
print  "Finding correlation matrix using heatmap:"                          True
```

The outliers in the dataset are identified and removed using the `RemoveOutlier()` and `DisplayOutlier()` functions.

pythonCopy code

```
print "Finding and removing outliers: " "Before removing Outliers:" print "Identifying
overall outliers in Column Name variables" "lstat" "rm"
"After removing Outliers"
```

The dataset is then split into input (`x`) and output (`y`) data.

pythonCopy code

```
"rm" "lstat" "medv"
```

The data is further split into training and testing sets using the `train_test_split()` function from scikit-learn.

pythonCopy code

```
from import
0.20 0
```

A linear regression model is applied to the training data using scikit-learn's `LinearRegression()`.

pythonCopy code

```
from import
```

The accuracy of the model is evaluated using the mean absolute error (`mean_absolute_error()`) and the model score (`score()`).

pythonCopy code

```
from import print "MAE:"
print "Model Score:"
```

Finally, the user can input features (number of rooms and percentage of lower status of the population) to predict the house price using the trained model.

pythonCopy code

```
print "Predict House Price by giving User Input:" 6 19
print "Prediction:"
```

The predicted house price is displayed as the output.

MY

import pandas as pd import seaborn as sns import numpy as np import matplotlib.pyplot as plt df = pd.read_csv(r"C:\Users\niraj\Downloads\Social_Network_Ads.csv") print(df) print(df.info()) print(df.size) print(df.shape) print(df.columns) print(df.head()) print(df.tail()) print(df.sample()) print(df.describe()) print(df.dtypes) print(df.isnull().sum()) print(df.isna().sum()) sns.heatmap(df.corr(), annot=True) plt.show() l = ['Age', 'EstimatedSalary', 'Purchased'] for i in l: sns.boxplot(df[i]) plt.show() x = df[['Age', 'EstimatedSalary']] y = df['Purchased'] from sklearn.model_selection import train_test_split x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, random_state=0) from sklearn.preprocessing import StandardScaler x_train = StandardScaler().fit_transform(x_train) x_test = StandardScaler().fit_transform(x_test) from sklearn.linear_model import LogisticRegression model = LogisticRegression().fit(x_train, y_train) y_pred = model.predict(x_test) from sklearn.metrics import classification_report print(classification_report(y_test, y_pred)) from sklearn.metrics import confusion_matrix cm = confusion_matrix(y_test, y_pred) print(cm) sns.heatmap(cm, annot=True, linewidths=.3) plt.show()

ChatGPT

The code you provided performs the following steps:

1. The required libraries are imported: pandas, seaborn, numpy, and matplotlib.pyplot.

pythonCopy code

```
import        as    import        as    import         as    import                 as
```

2. The dataset is read from a CSV file called "Social_Network_Ads.csv" using pandas' `read_csv()` function, and it is stored in a DataFrame called `df`.

pythonCopy code

```
              r"C:\Users\niraj\Downloads\Social_Network_Ads.csv"
```

3. Various operations are performed on the DataFrame to gather information about the dataset:
   - `info()` displays the summary information of the DataFrame, including the data types and non-null counts of each column.
   - `size` returns the total number of elements in the DataFrame.
   - `shape` returns the dimensions (rows, columns) of the DataFrame.
   - `columns` returns the names of the columns.
   - `head()` displays the first few rows of the DataFrame.
   - `tail()` displays the last few rows of the DataFrame.
   - `sample()` displays a random sample of rows from the DataFrame.
   - `describe()` provides descriptive statistics of the numeric columns in the DataFrame.
   - `dtypes` returns the data types of each column.
   - `isnull().sum()` and `isna().sum()` return the count of missing values in each column.

pythonCopy code

```
print          print       print          print              print          print
print             print               print          print        sum    print        sum
```

4. A heatmap is created using seaborn's `heatmap()` function to visualize the correlation between the columns of the DataFrame.

pythonCopy code

```
                          True
```

5. Box plots are created for the columns "Age", "EstimatedSalary", and "Purchased" using seaborn's `boxplot()` function.

pythonCopy code

```
   'Age'  'EstimatedSalary'  'Purchased'  for  in
```

6. The dataset is split into input (`x`) and output (`y`) data for training and testing a logistic regression model. The data is also scaled using StandardScaler from scikit-learn.

pythonCopy code

```
     'Age'  'EstimatedSalary'       'Purchased'  from                   import
                                                     0.1          0  from
                  import

```

7. Logistic regression is applied to the training data using scikit-learn's `LogisticRegression()`.

pythonCopy code

```
from                 import
```

8. Predictions are made on the test data using the trained model.

pythonCopy code

```

```

9. Classification report and confusion matrix are generated to evaluate the model's performance.

pythonCopy code

```
from            import                              print
                                    print                       True          .3

```

The classification report provides metrics such as precision, recall, and F1-score for each class. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives. Finally, a heatmap is created to visualize the confusion matrix.

MY

#Assignment No. 6 Classification on IRIS Dataset #Program by Prof. Rahul Dagade & Prof. Varsha Nale def RemoveOutlier(df, var): Q1 = df[var].quantile(0.25) Q3 = df[var].quantile(0.75) IQR = Q3 - Q1 high, low = Q3 + 1.5 * IQR, Q1 - 1.5 + IQR df = df[((df[var] >= low) & (df[var] <= high))] return df def DisplayOutlier(df, msg): fig,axes = plt.subplots(2, 2) fig.suptitle(msg) sns.boxplot(data=df, x="sepal.length", ax=axes[0,0]) sns.boxplot(data=df, x="sepal.width", ax=axes[0,1]) sns.boxplot(data=df, x="petal.length", ax=axes[1,0]) sns.boxplot(data=df, x="petal.width", ax=axes[1,1]) fig.tight_layout() plt.show() # import libraries import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt # Read Dataset df = pd.read_csv(r"C:\Users\niraj\Downloads\iris - 2023-05-20T153708.481.csv") print("Iris dataset is successfully loaded") # Display information of dataset print("Information of Dataset:\n", df.info) print("Shape of Dataset row x column):", df.shape) print("Columns Name:", df.columns) print("Total elements in dataset:", df.size) print("Datatype of attributes (columns):", df.dtypes) print("First 5 rows: \n", df.head().T) print("last 5 rows:\n", df.tail().T) print("Any 5 rows: \n", df.sample(5).T) # Find missing values print("Missing values") print(df.isnull().sum()) #Finding and removing outliers print("Finding and removing outliers: ") DisplayOutlier(df, "Before removing Outliers") df = RemoveOutlier(df, "sepal.length") df = RemoveOutlier(df, "sepal.width") df = RemoveOutlier(df, "petal.length") df = RemoveOutlier(df, "petal.width") DisplayOutlier(df,"After removing Outliers") #Encoding of output variable df["variety"]=df["variety"].astype('category') df["variety"]=df["variety"].cat.codes #Find correlation matrix- print("Finding correlation matrix using heatmap:") sns.heatmap(df.corr(),annot=True) plt.show() #Split the data into inputs and outputs x= df.iloc[:, [0,1,2,3]].values y= df.iloc[:, 4].values #Training and testing data from sklearn.model_selection import train_test_split #Assign test data size 20% x_train, x_test, y_train, y_test =train_test_split(x,y,test_size=0.20, random_state=0) #Normalization of input data from sklearn.preprocessing import StandardScaler sc_X = StandardScaler() x_train= sc_X.fit_transform(x_train) x_test= sc_X.fit_transform(x_test) #Apply Gaussian Naive Bayes classifier from sklearn.naive_bayes import GaussianNB model=GaussianNB() model.fit(x_train, y_train) y_pred=model.predict(x_test) #Display classification report from sklearn.metrics import classification_report print(classification_report(y_test, y_pred)) #Display confusion matrix from sklearn.metrics import confusion_matrix cm = confusion_matrix(y_test, y_pred) print("confusion matrix\n",cm) fig, ax=plt.subplots(figsize=(5, 5)) sns.heatmap(cm,annot=True,linewidths=.3,cmap="Blues") plt.show()

ChatGPT

2 / 2

The code you provided performs the following steps for classification on the IRIS dataset:

1. The required libraries are imported: pandas, seaborn, numpy, and matplotlib.pyplot.

pythonCopy code

```
import          as     import          as     import          as     import                  as
```

2. The dataset is read from a CSV file called "iris - 2023-05-20T153708.481.csv" using pandas' `read_csv()` function, and it is stored in a DataFrame called `df`.

pythonCopy code

```
                r"C:\Users\niraj\Downloads\iris - 2023-05-20T153708.481.csv"
```

3. Various operations are performed on the DataFrame to gather information about the dataset:
    - `info()` displays the summary information of the DataFrame, including the data types and non-null counts of each column.
    - `shape` returns the dimensions (rows, columns) of the DataFrame.
    - `columns` returns the names of the columns.
    - `size` returns the total number of elements in the DataFrame.
    - `dtypes` returns the data types of each column.
    - `head()` displays the first few rows of the DataFrame.
    - `tail()` displays the last few rows of the DataFrame.
    - `sample()` displays a random sample of rows from the DataFrame.
    - `isnull().sum()` returns the count of missing values in each column.

pythonCopy code

```
print "Information of Dataset:\n"            print "Shape of Dataset row x column):"
print "Columns Name:"            print "Total elements in dataset:"            print "Datatype of
```

```
attributes (columns):"          print  "First 5 rows: \n"          print  "last 5 rows:\n"
print  "Any 5 rows: \n"          5     print  "Missing values"  print          sum
```

4. The function `DisplayOutlier()` is defined to create box plots for the features ("sepal.length", "sepal.width", "petal.length", "petal.width") before and after removing outliers.

pythonCopy code

```
def DisplayOutlier                               2  2
  "sepal.length"          0  0                   "sepal.width"          0  1
  "petal.length"          1  0                   "petal.width"          1  1
```

5. The function `RemoveOutlier()` is defined to remove outliers from the DataFrame based on the specified variable.

pythonCopy code

```
def RemoveOutlier                        0.25                   0.75
            1.5          1.5                                   return
```

6. Outliers are displayed before and after removal using the `DisplayOutlier()` function.

pythonCopy code

```
print  "Finding and removing outliers: "          "Before removing Outliers"
                 "sepal.length"                   "sepal.width"
"petal.length"                 "petal.width"                 "After removing Outliers"
```

7. The output variable "variety" is encoded using categorical encoding, converting it into numerical form.

pythonCopy code

```
  "variety"      "variety"       'category'    "variety"      "variety"
```

8. The correlation matrix of the DataFrame is computed using `heatmap()` from seaborn library and displayed using `plt.show()`.

pythonCopy code

```
                          True
```

9. The input features and output variable are separated into `x` and `y`, respectively.

pythonCopy code

```
                0  1  2  3               4
```

10. The data is split into training and testing sets using `train_test_split()` from sklearn.

pythonCopy code

```
from                   import
             0.20               0
```

11. The input data is normalized using `StandardScaler()` from sklearn.

pythonCopy code

```
from                  import
```

12. Gaussian Naive Bayes classifier is applied to the training data using `GaussianNB()` from sklearn.

pythonCopy code

```
from                 import
```

13. The model is used to predict the output for the test data.

pythonCopy code

14. Classification report and confusion matrix are displayed using `classification_report()` and `confusion_matrix()` from sklearn, respectively.

pythonCopy code

```
from import print
print "confusion matrix\n"
5 5 True .3 "Blues"
```

This program performs classification on the IRIS dataset, removes outliers, encodes the output variable, normalizes the input data, and applies the Gaussian Naive Bayes classifier to make predictions. Finally, it displays the classification report and confusion matrix to evaluate the performance of the model.

MY

import nltk import re import pandas as pd from sklearn.feature_extraction.text import TfidfVectorizer from nltk.tokenize import sent_tokenize, word_tokenize from nltk.corpus import stopwords from nltk.stem import PorterStemmer, WordNetLemmatizer nltk.download("punkt") nltk.download("stopwords") nltk.download("wordnet") nltk.download("averaged_perceptron_tagger") nltk.download("omw-1.4") # Sentence Tokenization text = "Tokenization is the first step in text analytics. The process of breaking down a text par" tokenized_text = sent_tokenize(text) print(tokenized_text) # Word Tokenization tokenized_word = word_tokenize(text) print(tokenized_word) # Print stop words of English stop_words = set(stopwords.words("english")) print(stop_words) # Remove stop words from text text = "How to remove stop words with NLTK Library in Python?" text = re.sub("[^a-zA-Z]", " ", text) tokens = word_tokenize(text.lower()) filtered_text = [w for w in tokens if w not in stop_words] print("Tokenized Sentence:", tokens) print("Filtered Sentence:", filtered_text) # Stemming e_words = ["wait", "waiting", "waited", "waits"] ps = PorterStemmer() stemmed_words = [ps.stem(w) for w in e_words] print(stemmed_words) # Lemmatization wordnet_lemmatizer = WordNetLemmatizer() text = "studies studying cries cry" tokenization = nltk.word_tokenize(text) lemmatized_words = [wordnet_lemmatizer.lemmatize(w) for w in tokenization] print("Lemmatized Words:", lemmatized_words) # Part-of-Speech Tagging data = "The pink sweater fit her perfectly" words = word_tokenize(data) pos_tags = nltk.pos_tag(words) print(pos_tags) # TF-IDF Calculation d0 = "Jupiter is the largest Planet" d1 = "Mars is the fourth planet from the Sun" string = [d0, d1] data = pd.Series(string) tfidf = TfidfVectorizer() result = tfidf.fit_transform(data) print("Word indices:", tfidf.vocabulary_) print("TF-IDF Values:", result)

ChatGPT

The code you provided performs the following operations using the NLTK library and scikit-learn's TfidfVectorizer:

1. The required libraries and resources from NLTK are imported: nltk, re, pandas, TfidfVectorizer, sent_tokenize, word_tokenize, stopwords, PorterStemmer, and WordNetLemmatizer.

pythonCopy code

```
import import import as from import
from import from import from
import "punkt"
```

```
"stopwords" "wordnet" "averaged_perceptron_tagger"
"omw-1.4"
```

2. Sentence tokenization is performed using `sent_tokenize()` from NLTK.

pythonCopy code

```
"Tokenization is the first step in text analytics. The process of breaking down a text par"
print
```

3. Word tokenization is performed using `word_tokenize()` from NLTK.

pythonCopy code

```
print
```

4. The stop words for the English language are printed using `stopwords.words("english")` from NLTK.

pythonCopy code

```
set "english" print
```

5. Stop words are removed from a given text using regular expressions and list comprehension.

pythonCopy code

```
"How to remove stop words with NLTK Library in Python?" "[^a-zA-Z]" " "
for in if not in print "Tokenized
Sentence:" print "Filtered Sentence:"
```

6. Stemming is applied to a list of words using the PorterStemmer from NLTK.

pythonCopy code

```
"wait" "waiting" "waited" "waits" for
in print
```

7. Lemmatization is performed on a sentence using WordNetLemmatizer from NLTK.

pythonCopy code

```
"studies studying cries cry"
for in
print "Lemmatized Words:"
```

8. Part-of-speech tagging is performed on a sentence using `pos_tag()` from NLTK.

pythonCopy code

```
"The pink sweater fit her perfectly"
print
```

9. TF-IDF calculation is performed using scikit-learn's TfidfVectorizer.

pythonCopy code

```
"Jupiter is the largest Planet" "Mars is the fourth planet from the Sun"
print "Word indices:"
print "TF-IDF Values:"
```

This code showcases various text preprocessing techniques such as sentence tokenization, word tokenization, stop word removal, stemming, lemmatization, part-of-speech tagging, and TF-IDF calculation using NLTK and scikit-learn libraries.

MY

```
#Assignment no. 8 Data Visualization on Titanic Dataset (Histogram) # Program by Prof. Rahul
Dagade & Prof. Varsha Nale #import libraries import numpy as np import pandas as pd import
seaborn as sns import matplotlib.pyplot as plt #Read Dataset df=pd.read_csv(r"titanic.csv")
print("Titanic dataset is successfully loaded into Data Frame....") #Display information of dataset print(
"Information of Dataset: \n", df.info) print( "Shape of Dataset (row x column):",df.shape) print(
"Columns Name:", df.columns) print( "Total elements in dataset:", df.size) print( "Datatype of
attributes (columns):", df.dtypes) print( "First 5 rows: \n", df.head().T) print( "Last 5 rows:\n",df.tail().T)
print( "Any 5 rows: \n",df.sample(5).T) #Find missing values print ("Missing values")
print(df.isnull().sum()) #Fill the missing values df ["Age"].fillna (df["Age"].median(), inplace=True) print
("Null values are: \n",df.isnull().sum()) #Histogram of 1-variable fig, axes= plt.subplots(1,2) fig.
suptitle(" Histogram 1-variables (Age & Fare)") sns. histplot(data=df, x="Age", ax=axes[0]) sns.
histplot(data=df, x="Fare", ax=axes [1]) plt.show() #Histogram of 2-variables fig, axes
=plt.subplots(2,2) fig.suptitle( "Histogram of 2-variables") sns.histplot(data = df,
x="Age",hue="Survived", multiple="dodge", ax=axes[0,0]) sns.histplot(data = df,
x="Fare",hue="Survived", multiple="dodge", ax=axes[0,1]) sns.histplot(data = df, x="Age",hue="Sex",
multiple= "dodge", ax=axes[1,0]) sns.histplot(data = df, x="Fare", hue="Sex", multiple="dodge",
ax=axes [1, 1]) plt.show()
```

ChatGPT

The code provided performs the following operations for data visualization on the Titanic dataset using histograms:

1. Import the required libraries: numpy, pandas, seaborn, and matplotlib.

pythonCopy code

```
import          as     import          as     import          as     import                  as
```

2. Read the Titanic dataset into a DataFrame.

pythonCopy code

```
                   r"titanic.csv"   print   "Titanic dataset is successfully loaded into Data Frame...."
```

3. Display the information of the dataset using `info()`, the shape of the dataset using `shape`, column names using `columns`, total elements using `size`, data types of attributes using `dtypes`, and the first 5 rows using `head()`.

pythonCopy code

```
print  "Information of Dataset:\n"          print  "Shape of Dataset (row x column):"
print  "Columns Name:"           print  "Total elements in dataset:"        print  "Datatype of
attributes (columns):"         print  "First 5 rows:\n"
```

4. Find missing values in the dataset using `isnull().sum()`.

pythonCopy code

```
print  "Missing values"  print          sum
```

5. Fill the missing values in the "Age" column with the median value using `fillna()`.

pythonCopy code

```
   "Age"           "Age"                  True  print  "Null values are:\n"         sum
```

6. Create histograms for one variable, "Age" and "Fare", using `histplot()` from seaborn.

pythonCopy code

```
                  1  2          "Histogram 1-variables (Age & Fare)"
  "Age"         0                "Fare"           1
```

7. Create histograms for two variables, "Age" and "Fare", with the hue "Survived" and "Sex" using `histplot()` from seaborn.

pythonCopy code

```
2 2 "Histogram of 2-variables" "Age"
"Survived" "dodge" 0 0 "Fare" "Survived"
"dodge" 0 1 "Age" "Sex" "dodge" 1
0 "Fare" "Sex" "dodge" 1 1
```

This code generates histograms for the Titanic dataset, showcasing the distribution of variables such as age and fare. It also includes histograms with multiple variables and hue options, allowing for visual analysis of the relationship between variables and survival or gender.

MY

#Assignment no. 8 Data Visualization on Titanic Dataset (Histogram) # Program by Prof. Rahul Dagade & Prof. Varsha Nale #import libraries import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Read Dataset df=pd.read_csv(r"titanic.csv") print("Titanic dataset is successfully loaded into Data Frame....") #Display information of dataset print("Information of Dataset: \n", df.info) print("Shape of Dataset (row x column):",df.shape) print("Columns Name:", df.columns) print("Total elements in dataset:", df.size) print("Datatype of attributes (columns):", df.dtypes) print("First 5 rows: \n", df.head().T) print("Last 5 rows:\n",df.tail().T) print("Any 5 rows: \n",df.sample(5).T) #Find missing values print ("Missing values") print(df.isnull().sum()) #Fill the missing values df ["Age"].fillna (df["Age"].median(), inplace=True) print ("Null values are: \n",df.isnull().sum()) #Histogram of 1-variable fig, axes= plt.subplots(1,2) fig. suptitle(" Histogram 1-variables (Age & Fare)") sns. histplot(data=df, x="Age", ax=axes[0]) sns. histplot(data=df, x="Fare", ax=axes [1]) plt.show() #Histogram of 2-variables fig, axes =plt.subplots(2,2) fig.suptitle("Histogram of 2-variables") sns.histplot(data = df, x="Age",hue="Survived", multiple="dodge", ax=axes[0,0]) sns.histplot(data = df, x="Fare",hue="Survived", multiple="dodge", ax=axes[0,1]) sns.histplot(data = df, x="Age",hue="Sex", multiple= "dodge", ax=axes[1,0]) sns.histplot(data = df, x="Fare", hue="Sex", multiple="dodge", ax=axes [1, 1]) plt.show()

ChatGPT

The code provided performs the following operations for data visualization on the Titanic dataset using histograms:

1. Import the required libraries: numpy, pandas, seaborn, and matplotlib.

pythonCopy code

```
import as import as import as import as
```

2. Read the Titanic dataset into a DataFrame.

pythonCopy code

```
r"titanic.csv" print "Titanic dataset is successfully loaded into Data Frame...."
```

3. Display the information of the dataset using `info()`, the shape of the dataset using `shape`, column names using `columns`, total elements using `size`, data types of attributes using `dtypes`, and the first 5 rows using `head()`.

pythonCopy code

```
print "Information of Dataset:\n"          print "Shape of Dataset (row x column):"
print "Columns Name:"          print "Total elements in dataset:"          print "Datatype of
attributes (columns):"          print "First 5 rows:\n"
```

4. Find missing values in the dataset using `isnull().sum()`.

pythonCopy code

```
print "Missing values"  print          sum
```

5. Fill the missing values in the "Age" column with the median value using `fillna()`.

pythonCopy code

```
 "Age"          "Age"                    True  print "Null values are:\n"          sum
```

6. Create histograms for one variable, "Age" and "Fare", using `histplot()` from seaborn.

pythonCopy code

```
                    1  2          "Histogram 1-variables (Age & Fare)"
 "Age"          0                    "Fare"          1
```

7. Create histograms for two variables, "Age" and "Fare", with the hue "Survived" and "Sex" using `histplot()` from seaborn.

pythonCopy code

```
                    2  2          "Histogram of 2-variables"                    "Age"
     "Survived"          "dodge"          0  0                    "Fare"          "Survived"
          "dodge"          0  1                    "Age"          "Sex"          "dodge"          1
0                    "Fare"          "Sex"          "dodge"          1  1
```

This code generates histograms for the Titanic dataset, showcasing the distribution of variables such as age and fare. It also includes histograms with multiple variables and hue options, allowing for visual analysis of the relationship between variables and survival or gender.

# Assignment no. 9 Data Visualization on Titanic Dataset (Boxplot) #import libraries import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt #Read Dataset df=pd.read_csv(r"titanic.csv") print("Titanic dataset is successfully loaded into Data Frame....") #Display information of dataset print("Information of Dataset: \n", df.info) print("Shape of Dataset (row x column):",df.shape) print("Columns Name:", df.columns) print("Total elements in dataset:", df.size) print("Datatype of attributes (columns):", df.dtypes) print("First 5 rows: \n", df.head().T) print("Last 5 rows:\n",df.tail().T) print("Any 5 rows: \n",df.sample(5).T) #Find missing values print ("Missing values") print(df.isnull().sum()) #Fill the missing values df ["Age"].fillna (df["Age"].median(), inplace=True) print ("Null values are: \n",df.isnull().sum()) #Boxplot of 1-variable fig, axes =plt.subplots(1,2) fig.suptitle("Boxplot of 1-variables (Age & Fare)") sns.boxplot (data =df, x="Age", ax=axes[0]) sns.boxplot (data =df, x="Fare",ax=axes[1]) plt.show() #Boxplot of 2-variables fig, axes =plt.subplots(2,2) fig.suptitle("Boxplot of 2-variables") sns.boxplot (data = df, x="Survived", y="Age",

```
hue="Survived", ax=axes[0,0]) sns.boxplot (data = df, x="Survived", y="Fare",
hue="Survived",ax=axes[0,1]) sns.boxplot (data = df, x="Sex", y="Age", hue= "Sex", ax=axes[1,0])
sns.boxplot (data = df, x="Sex", y="Fare", hue="Sex", ax=axes[1,1]) plt.show() #Boxplot of 3-variables
fig, axes= plt.subplots(1,2) fig.suptitle("Boxplot of 3-variables") sns.boxplot (data=df, x="Sex",
y="Age", hue="Survived", ax=axes[0]) sns. boxplot(data=df, x = "Sex", y="Fare",hue="Survived",
ax=axes [1]) plt.show()
```

ChatGPT

The code provided performs data visualization on the Titanic dataset using boxplots. Here's a breakdown of the code:

1. Import the required libraries: numpy, pandas, seaborn, and matplotlib.

pythonCopy code

```
import          as    import          as    import          as    import                  as
```

2. Read the Titanic dataset into a DataFrame.

pythonCopy code

```
             r"titanic.csv"  print  "Titanic dataset is successfully loaded into Data Frame...."
```

3. Display the information of the dataset using `info()`, the shape of the dataset using `shape`, column names using `columns`, total elements using `size`, data types of attributes using `dtypes`, and the first 5 rows using `head()`.

pythonCopy code

```
print  "Information of Dataset:\n"        print  "Shape of Dataset (row x column):"
print  "Columns Name:"          print  "Total elements in dataset:"        print  "Datatype of
attributes (columns):"          print  "First 5 rows:\n"
```

4. Find missing values in the dataset using `isnull().sum()`.

pythonCopy code

```
print  "Missing values"  print          sum
```

5. Fill the missing values in the "Age" column with the median value using `fillna()`.

pythonCopy code

```
  "Age"          "Age"                True  print  "Null values are:\n"         sum
```

6. Create boxplots for one variable, "Age" and "Fare", using `boxplot()` from seaborn.

pythonCopy code

```
                 1  2          "Boxplot of 1-variables (Age & Fare)"
  "Age"         0                      "Fare"          1
```

7. Create boxplots for two variables, "Age" and "Fare", with the hue "Survived" and "Sex" using `boxplot()` from seaborn.

pythonCopy code

```
                 2  2          "Boxplot of 2-variables"                   "Survived"
  "Age"     "Survived"       0  0                  "Survived"    "Fare"     "Survived"
        0  1                     "Sex"   "Age"    "Sex"         1  0
  "Sex"   "Fare"      "Sex"        1  1
```

8. Create boxplots for three variables, "Sex", "Age", and "Fare", with the hue "Survived" using `boxplot()` from seaborn.

pythonCopy code

```
1 2 "Boxplot of 3-variables" "Sex" "Age"
"Survived" 0 "Sex" "Fare" "Survived" 1
```

These boxplots provide a visual representation of the distribution of variables in the Titanic dataset, allowing for the analysis of central tendency, variability, and potential outliers.

MY

```
# Assignment no, 10 Data Visualization on IRIS Dataset (Histogram & Boxplot) # import libraries
import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt
#Read Dataset df=pd.read_csv(r"iris - 2023-05-20T153708.481.csv") print("iris dataset is successfully
loaded into Data Frame....") #Display information of dataset print( "Information of Dataset: \n",
df.info) print( "Shape of Dataset (row x column):",df.shape) print( "Columns Name:", df.columns)
print( "Total elements in dataset:", df.size) print( "Datatype of attributes (columns):", df.dtypes) print(
"First 5 rows: \n", df.head().T) print( "Last 5 rows:\n",df.tail().T) print( "Any 5 rows: \n",df.sample(5).T)
#Find missing values print ("Missing values") print(df.isnull().sum()) #Histogram of 1-variable fig,
axes= plt.subplots(2,2) fig.suptitle("Histogram 1-variables") sns.histplot(data=df,x="sepal.length",
ax=axes[0,0]) sns.histplot(data=df,x="sepal.width", ax=axes [0,1])
sns.histplot(data=df,x="petal.length", ax=axes[1,0]) sns.histplot(data=df,x="petal.width",
ax=axes[1,1]) plt.show() #Histogram of 2-variables fig, axes =plt.subplots(2,2) fig.suptitle("Histogram
of 2-variables") sns.histplot(data=df, x="sepal.length",hue="variety", multiple="dodge", ax=axes[0,0])
sns.histplot(data=df, x="sepal.width",hue="variety", multiple="dodge", ax=axes[0,1])
sns.histplot(data=df, x="petal.length",hue="variety", multiple ="dodge",ax=axes[1,0])
sns.histplot(data=df, x="petal.width",hue="variety", multiple="dodge", ax=axes[1,1]) plt.show()
#Boxplot of 1-variable. fig, axes= plt.subplots(2,2) fig.suptitle("Boxplot of 1-variables")
sns.histplot(data=df,x="sepal.length", ax=axes[0,0]) sns.histplot(data=df,x="sepal.width", ax=axes
[0,1]) sns.histplot(data=df,x="petal.length", ax=axes[1,0]) sns.histplot(data=df,x="petal.width",
ax=axes[1,1]) plt.show() #Boxplot of 2-variables fig, axes =plt.subplots(2,2) fig.suptitle("Histogram of
2-variables") sns.histplot(data=df, x="sepal.length",y="variety", hue="variety", ax=axes[0,0])
sns.histplot(data=df, x="sepal.width",y="variety", hue="variety", ax=axes[0,1]) sns.histplot(data=df,
x="petal.length",y="variety", hue="variety",ax=axes[1,0]) sns.histplot(data=df,
x="petal.width",y="variety", hue="variety", ax=axes[1,1]) plt.show()
```

ChatGPT

The code provided performs data visualization on the Iris dataset using histograms and boxplots. Here's a breakdown of the code:

1. Import the required libraries: numpy, pandas, seaborn, and matplotlib.

pythonCopy code

```
import as import as import as import as
```

2. Read the Iris dataset into a DataFrame.

pythonCopy code

```
r"iris - 2023-05-20T153708.481.csv" print "Iris dataset is successfully loaded into Data
Frame...."
```

3. Display the information of the dataset using `info()`, the shape of the dataset using `shape`, column names using `columns`, total elements using `size`, data types of attributes using `dtypes`, and the first 5 rows using `head()`.

pythonCopy code

```
print "Information of Dataset:\n" print "Shape of Dataset (row x column):"
print "Columns Name:" print "Total elements in dataset:" print "Datatype of
attributes (columns):" print "First 5 rows:\n"
```

4. Find missing values in the dataset using `isnull().sum()`.

pythonCopy code

```
print "Missing values" print sum
```

5. Create histograms for one variable, "sepal.length", "sepal.width", "petal.length", and "petal.width" using `histplot()` from seaborn.

pythonCopy code

```
2 2 "Histogram 1-variables" "sepal.length"
0 0 "sepal.width" 0 1
"petal.length" 1 0 "petal.width" 1 1
```

6. Create histograms for two variables, "sepal.length", "sepal.width", "petal.length", and "petal.width", with the hue "variety" using `histplot()` from seaborn.

pythonCopy code

```
2 2 "Histogram of 2-variables" "sepal.length"
"variety" "dodge" 0 0 "sepal.width" "variety"
"dodge" 0 1 "petal.length" "variety" "dodge"
1 0 "petal.width" "variety" "dodge" 1 1
```

7. Create boxplots for one variable, "sepal.length", "sepal.width", "petal.length", and "petal.width" using `boxplot()` from seaborn.

pythonCopy code

```
2 2 "Boxplot of 1-variables" "sepal.length"
0 0 "sepal.width" 0 1 "petal.length"
1 0 "petal.width" 1 1
```

8. Create boxplots for two variables, "sepal.length", "sepal.width", "petal.length", and "petal.width", with the hue "variety" using `boxplot()` from seaborn.

pythonCopy code

```
2 2 "Boxplot of 2-variables" "sepal.length"
"variety" "variety" 0 0 "sepal.width" "variety"
"variety" 0 1 "petal.length" "variety" "variety"
1 0 "petal.width" "variety" "variety" 1 1
```

These visualizations provide insights into the distribution and variation of the Iris dataset using histograms and boxplots. They help in understanding the characteristics and relationships between different variables in the dataset.