CONTENTS

Practical 1 Find the below data set and perform the following operations:-

Dataset name: -mtcars_DataDescription

1. Read the dataset.
2. Find the head of the dataset.
3. Find the Datatype of Dataset (each column).
4. From the given dataset 'mtcars.csv', plot a histogram to check the frequency distribution of the variable 'mpg' (Miles per gallon).
5. Find the highest frequency of interval.
6. Which can be inferred from scatter plot of 'mpg' (Miles per gallon) vs 'wt' (Weight of car) from the dataset mtcars.csv.

Step 1: Read the dataset

```
import pandas as pd

# Assuming the dataset 'mtcars.csv' is in the current directory
mtcars = pd.read_csv('mtcars.csv')
```

Step 2: Find the head of the dataset
To examine the first few rows of the dataset:

```
print(mtcars.head())
```

Step 3: Find the Datatype of Dataset (each column)
To determine the datatype of each column:

```
print(mtcars.dtypes)
```

Step 4: Plot a histogram of the 'mpg' variable
To visualize the frequency distribution of the 'mpg' (Miles per gallon) variable using a histogram:

```
import matplotlib.pyplot as plt
import pandas as pd

mtcars = pd.read_csv('mtcars.csv')
plt.hist(mtcars['mpg'], bins=10, edgecolor='black')
plt.xlabel('Miles per gallon (mpg)')
plt.ylabel('Frequency')
plt.title('Histogram of Miles per gallon (mpg)')
plt.show()
```

Step 5: Find the highest frequency of interval

```
import matplotlib.pyplot as plt
import numpy as np

# Generate some example data
data = np.random.normal(loc=0, scale=1, size=1000)

# Create a histogram
plt.hist(data, bins=10, edgecolor='black')
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Histogram of Data')
plt.grid(True)
```

```
# Display the plot
plt.show()



# Get the histogram data
counts, bins, _ = plt.hist(data, bins=10, edgecolor='black')

# Find the bin with the maximum frequency
max_frequency = np.max(counts)
max_index = np.argmax(counts)
max_interval = (bins[max_index], bins[max_index + 1])

print(f"The interval with the highest frequency is {max_interval} with frequency
{max_frequency}.")
```

Step 6: Inference from scatter plot of 'mpg' vs 'wt'

To infer from the scatter plot of 'mpg' (Miles per gallon) vs 'wt' (Weight of car), we would typically look for a trend or pattern that shows how the mileage (mpg) changes with the weight (wt) of the car. Common inferences could include:

Negative correlation: If the scatter plot shows a downward trend (as weight increases, mpg decreases), it suggests that heavier cars tend to have lower mileage.

No correlation: If there's no clear pattern (scatter points are randomly distributed), then there might not be a strong relationship between weight and mileage.

Positive correlation: In some cases, a positive trend (as weight increases, mpg increases) could imply that heavier cars are more fuel-efficient, but this is less common for standard combustion engine cars.

Here's how you could create a scatter plot to observe this relationship:

```
plt.scatter(mtcars['wt'], mtcars['mpg'], color='blue')
plt.xlabel('Weight of car (wt)')
plt.ylabel('Miles per gallon (mpg)')
plt.title('Scatter plot of Miles per gallon vs Weight of car')
plt.show()
```

After generating this scatter plot, observe the direction of the relationship between 'mpg' and 'wt' to make your inference.

Practical 2 : Find the below data set and perform the following operations: Dataset name: - Churn_DataDescription

1. Find the no. of duplicate records in the churn dataframe based on the cutomerID column.

2. In the churn dataframe, what are the total no. of missing values for the variable TotalCharges?

3. From the churn dataframe, what is the average monthly charge paid by a customer for the services he/she has signed up for?

4. In the churn dataframe, under the variable Dependents how many records have "1@#" ?

5. Find the data type of the variable tenure from the churn dataframe.

1. Number of duplicate records based on CustomerID column:

```
import pandas as pd
# Load the dataset
churn = pd.read_csv('Churn_DataDescription.csv')
# Find duplicate records based on CustomerID column
duplicate_records = churn.duplicated(subset=['CustomerID']).sum()

print(f"Number of duplicate records based on CustomerID: {duplicate_records}")
```

2. Total number of missing values for the variable TotalCharges:

```
# Find total number of missing values for TotalCharges
missing_total_charges = churn['TotalCharges'].isnull().sum()
print(f"Total number of missing values for TotalCharges: {missing_total_charges}")
```

3. Average monthly charge paid by a customer

```
Assuming the column representing monthly charges is named MonthlyCharges:
# Calculate average monthly charge
average_monthly_charge = churn['MonthlyCharges'].mean()
print(f"Average monthly charge paid by a customer: {average_monthly_charge}")
```

4. Number of records in the Dependents column that have "1@#":

```
# Count records in Dependents column that have "1@#"
dependents_count = (churn['Dependents'] == '1@#').sum()
print(f"Number of records in Dependents column with '1@#': {dependents_count}")
```

5. Find the data type of the variable tenure from the churn dataframe

```
# Check the data type of the variable 'tenure'
tenure_dtype = churn['tenure'].dtype print(f"Data type of the variable 'tenure': {tenure_dtype}")
```

Practical 3 Find the below data set and perform the following operations:

Dataset name: -Diamond_DataDescription

1. Plot a boxplot for "price" vs "cut" from the dataset "diamond.csv". Which of the categories under "cut" have the highest median price?

2. Create a frequency table (one-way table) for the variable "cut" from the dataset "diamond.csv". What is the frequency for the cut type "Ideal"?

3. Show the subplot of the diamond carat weight distribution.

4. Show the subplot of diamond depth distribution.

5. Build the Model using linear regression and find the accuracy.

Step 1 Load data set

```
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset
diamonds = pd.read_csv('Diamonds_DataDescription.csv')
```

1. Plot a boxplot for "price" vs "cut" from the dataset "diamond.csv". Which of the categories under "cut" have the highest median price?

```
# Boxplot for price vs cut
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.boxplot(x='cut', y='price', data=diamonds, order=['Fair', 'Good', 'Very Good', 'Premium', 'Ideal'])
plt.title('Boxplot of Price vs Cut')
plt.xlabel('Cut')
plt.ylabel('Price')
plt.show()
```

2. Create a frequency table (one-way table) for the variable "cut" from the dataset "diamond.csv". What is the frequency for the cut type "Ideal"?

```
# Frequency table for cut
cut_frequency = diamonds['cut'].value_counts()
print(cut_frequency)
# Frequency for cut type 'Ideal'
frequency_ideal = cut_frequency['Ideal']
print(frequency_ideal)
```

3. Show the subplot of the diamond carat weight distribution.

```
# Subplot of diamond carat weight distribution
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.hist(diamonds['carat'], bins=30, color='skyblue', edgecolor='black')
plt.title('Diamond Carat Weight Distribution')
plt.xlabel('Carat')
plt.ylabel('Frequency')
plt.show()
```

4. Show the subplot of diamond depth distribution.

```
# Subplot of diamond depth distribution
plt.subplot(1, 2, 2)
plt.hist(diamonds['depth'], bins=30, color='lightgreen', edgecolor='black')
plt.title('Diamond Depth Distribution')
plt.xlabel('Depth')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

5. Build the Model using linear regression and find the accuracy.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Extract features and target
X = diamonds[['carat', 'depth']]
y = diamonds['price']
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize Linear Regression model
model = LinearRegression()
# Fit the model
model.fit(X_train, y_train)
# Predict on the test set
y_pred = model.predict(X_test)
# Calculate R-squared
accuracy = r2_score(y_test, y_pred)
print(f'Accuracy (R-squared): {accuracy}')plt.tight_layout()
plt.show()
```

Practical 4 Use the dataset named "People Charm case.csv" that deals with HR analytics.
Answer the following questions:

1. Which of the variables have missing values?
2. What is the third quartile value for the variable "lastEvaluvation"?
3. Construct a Crosstable for the variables 'dept' and "salary" and find out which department has highest frequency value in the category low salary.
4. Generate a boxplot for the variable "numberOfProjects" and get the median value for the number of projects where the employees have worked on.
5. Plot a histogram using the variable "avgMonthlyHours" and find the range in which the number of employees worked for 150 hours per month?
6. Generate a boxplot for the variables "lastEvaluation" and "numberOfProjects"

1. Which of the variables have missing values?

```
import pandas as pd
# Load the dataset
df = pd.read_csv("People Charm case.csv")
# Check for missing values
missing_values = df.isnull().sum()
variables_with_missing = missing_values[missing_values > 0].index.tolist()
print("Variables with missing values:")
print(variables_with_missing)
```

2. What is the third quartile value for the variable "lastEvaluvation"?

```
# Assuming 'lastEvaluvation' is a typo and should be 'lastEvaluation'
third_quartile = df['lastEvaluation'].quantile(0.75)
print("Third quartile value for lastEvaluation:", third_quartile)
```

3. Construct a Crosstable for the variables 'dept' and "salary" and find out which department has highest frequency value in the category low salary.

```
# Crosstab and finding the department with highest frequency of low salary
cross_table = pd.crosstab(df['dept'], df['salary'])

# Find department with highest frequency of low salary
dept_with_highest_low_salary = cross_table['low'].idxmax()
highest_freq_low_salary = cross_table.loc[dept_with_highest_low_salary, 'low']

print("Department with highest frequency of low salary:", dept_with_highest_low_salary)
print("Frequency of low salary:", highest_freq_low_salary)
```

4. Generate a boxplot for the variable “numberOfProjects” and get the median value for the number of projects where the employees have worked on.

```
import matplotlib.pyplot as plt

# Boxplot for numberOfProjects
plt.figure(figsize=(8, 6))
plt.boxplot(df['numberOfProjects'])
plt.title('Boxplot of numberOfProjects')
plt.ylabel('Number of Projects')
plt.show()

# Median value
median_projects = df['numberOfProjects'].median()
print("Median number of projects:", median_projects)
```

5. Plot a histogram using the variable “avgMonthlyHours” and find the range in which the number of employees worked for 150 hours per month?

```
# Histogram for avgMonthlyHours
plt.figure(figsize=(8, 6))
plt.hist(df['avgMonthlyHours'], bins=20, edgecolor='black')
plt.title('Histogram of avgMonthlyHours')
plt.xlabel('Average Monthly Hours')
plt.ylabel('Frequency')
plt.show()

# Range for 150 hours
num_employees_150_hours = ((df['avgMonthlyHours'] >= 150) & (df['avgMonthlyHours'] <
160)).sum()
print("Number of employees worked 150 hours per month:", num_employees_150_hours)
```

6. Generate a boxplot for the variables “lastEvaluation” and “numberOfProjects”

```
import seaborn as sns

# Boxplot for lastEvaluation and numberOfProjects
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[['lastEvaluation', 'numberOfProjects']])
plt.title('Boxplot of lastEvaluation and numberOfProjects')
plt.ylabel('Value')
plt.show()
```

Practical 5 : Use the dataset named "People Charm case.csv" that deals with HR analytics and answer the following questions:

1. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as The accuracy score for the predicted model is?

2. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as 2 and find out how many samples are misclassified?

3. Build a k-Nearest Neighbors model using all the variables. Use 75% of the data as the training set, fix the random state as 0 and the k value as 2.The accuracy score for the predicted model is?

##Use the dataset named "People Charm case.csv" that deals with HR analytics and answer the following questions

1. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as The accuracy score for the predicted model is?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv('Charm.csv')

# Define features and target
X = df.drop(columns='numberOfProjects')  # Replace 'numberOfProjects' with your actual target column name
y = df['numberOfProjects']  # Replace 'numberOfProjects' with your actual target column name

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Create a preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), X.select_dtypes(include=['int64', 'float64']).columns),
        ('cat', OneHotEncoder(), X.select_dtypes(include=['object']).columns)
    ]
)

# Create a logistic regression model with increased iterations
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42, max_iter=1000))  # Increase max_iter to 1000
])

# Train the model
model.fit(X_train, y_train)
```

```
# Predict on the test set
y_pred = model.predict(X_test)
# Evaluate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy score: {accuracy:.4f}')
```

2. Build a Logistic Regression model using all the variables. Use 75% of the data as the training set and fix the random state as 2 and find out how many samples are misclassified?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import confusion_matrix
import numpy as np

# Load the dataset
# Replace this with the actual dataset path or method to load your data
data = pd.read_csv('Charm.csv')  # Replace with your actual file path

# Define your features (X) and target (y)
X = data.drop(columns=['numberOfProjects'])  # Replace 'numberOfProjects' with the actual target column name
y = data['numberOfProjects']  # Replace 'numberOfProjects' with the actual target column name

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Define the preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
    ],
    remainder='passthrough'  # Keep other columns as they are
)

# Define the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000, random_state=2))  # Increased max_iter to 1000
])

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=2)

# Train the model
pipeline.fit(X_train, y_train)

# Make predictions
y_pred = pipeline.predict(X_test)
```

```
# Calculate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print confusion matrix for verification
print("Confusion Matrix:")
print(cm)

# Calculate the number of misclassified samples
misclassified_samples = np.sum(cm) - np.trace(cm)

print(f"Number of misclassified samples: {misclassified_samples}")
```

3. Build a k-Nearest Neighbors model using all the variables. Use 75% of the data as the training set, fix the random state as 0 and the k value as 2.The accuracy score for the predicted model is?

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Load your dataset (replace 'your_data.csv' with your actual file)
data = pd.read_csv('People Charm case.csv')

# Define your features (X) and target (y)
X = data.drop(columns=['numberOfProjects'])  # Replace 'target_column' with the actual target
column name
y = data['numberOfProjects']  # Replace 'target_column' with the actual target column name

# Identify categorical columns
categorical_columns = X.select_dtypes(include=['object']).columns

# Define the preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns)
    ],
    remainder='passthrough'  # Keep other columns as they are
)
# Define the pipeline
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', KNeighborsClassifier(n_neighbors=2, algorithm='auto'))  # k=2
])
```

```
# Split the data into training and testing sets (75% training, 25% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
# Train the model
pipeline.fit(X_train, y_train)
# Make predictions
y_pred = pipeline.predict(X_test)

# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy score: {accuracy:.2f}")
```

Practical 6 - Data from an online microlending platform has been collected. This data contains details of the purpose for which the loans would be used and how the loan is funded. Additional information on the country of loan recipient and the poverty levels of the country are also given.
It is to be seen whether a loan would be funded or not based on the available data.

Variable Description:

Parameter	Description
activity	Activity for which loan was requested
borrower_genders	Gender of the borrowers
country	Country in which loan was disbursed
country_code	ISO country code
currency_policy	The currency policy in which loan was disbursed
distribution_model	Loan disbursed through field partner or not
lender_count	the total number of lenders that contributed to this loan
original_language	language of the original loan application
loan_amount	The amount disbursed by the field agent to the borrower(USD)
repayment_interval	intervel between payments
sector	High level category
status	The status of a loan : whether funded,not funded
term_in_months	The duration for which the loan was disbursed in months
rMPI	Multiple Poverty Index

1. How many columns are of 'object' data type? Read the given data "lendingdata.csv" and save it as a dataframe called data, and answer the questions below:

2. Find the total number of missing values in the data set?

3. Identify which of the columns contain redundant information and can be dropped from the dataframe.

4. What is the third quartile value of the variable "loan_amount"?

5. What is the percentage split of the different categories in the column "repayment_interval" after dropping the missing values?

6. What is the minimum loan amount disbursed in the Agriculture sector?

Solution :

```
# Import necessary libraries
import pandas as pd
# Load the data into a dataframe
data = pd.read_csv('lendingdata.csv')
# Step 1: Determine the number of columns with 'object' data type
object_columns = data.select_dtypes(include=['object']).columns
num_object_columns = len(object_columns)
print(f"Number of columns with 'object' data type: {num_object_columns}")
print(f"Columns with 'object' data type: {object_columns.tolist()}")

# Step 2: Find the total number of missing values in the dataset
total_missing_values = data.isnull().sum().sum()
print(f"Total number of missing values in the data set: {total_missing_values}")

# Step 3: Identify and drop redundant columns
# Assuming 'country_code' is redundant because 'country' is present
columns_to_drop = ['country_code']
data = data.drop(columns=columns_to_drop)
print(f"Columns after dropping redundant information: {data.columns.tolist()}")

# Step 4: Calculate the third quartile value (Q3) of the variable 'loan_amount'
third_quartile_value = data['loan_amount'].quantile(0.75)
print(f"Third quartile value of the variable 'loan_amount': {third_quartile_value}")

# Step 5: Calculate the percentage split of different categories in 'repayment_interval'
# Drop rows with missing values in 'repayment_interval'
repayment_data = data['repayment_interval'].dropna()

# Calculate the percentage split of different categories
percentage_split = repayment_data.value_counts(normalize=True) * 100
print("Percentage split of different categories in 'repayment_interval':")
print(percentage_split)

# Step 6: Find the minimum loan amount disbursed in the Agriculture sector
min_loan_amount_agriculture = data[data['sector'] == 'Agriculture']['loan_amount'].min()
print(f"Minimum loan amount disbursed in the Agriculture sector: {min_loan_amount_agriculture}")
```

Practical 7 - Sentiment analysis, also known as opinion mining, is the process of determining the emotional tone behind a series of words. It is a common NLP task used to identify the sentiment of texts like reviews, social media posts, or news articles. Below is a simple Python program that performs sentiment analysis using the NLTK library and the VADER sentiment analysis tool.

Solution :

```
# Import necessary libraries
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
# Download necessary NLTK resources
nltk.download('vader_lexicon')
# Initialize the SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
# Example texts for sentiment analysis
texts = [
    "I love this product! It's absolutely amazing and worth every penny.",
    "This is the worst experience I've ever had. I will never come back.",
    "The movie was okay, not great but not terrible either.",
    "I am very happy with the service provided, totally satisfied.",
    "I'm disappointed. The quality is not what I expected.",
    "The book is interesting, but it has some dull moments.",
    "The support team is helpful and resolved my issue quickly."
]

# Function to analyze sentiment
def analyze_sentiment(text):
    sentiment = sia.polarity_scores(text)
    overall_sentiment = 'Positive' if sentiment['compound'] > 0 else 'Negative' if
sentiment['compound'] < 0 else 'Neutral'
    return sentiment, overall_sentiment

# Perform sentiment analysis on each text
for text in texts:
    print(f"Text: {text}")
    sentiment_scores, overall_sentiment = analyze_sentiment(text)
    print(f"Sentiment Scores: {sentiment_scores}")
    print(f"Overall Sentiment: {overall_sentiment}\n")
```