Data Collection and Preprocessing Phase

Date	15 March 2024
Team ID	SWTID1720014456
Project Title	Thyroid Classification
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	Description
Data Overview	• **Basic Statistics:** Calculate mean, median, standard deviation, minimum, and maximum for each feature (e.g., thyroid hormone levels, age, etc.). • **Dimensions:** Identify the number of samples (rows) and features (columns) in the dataset. • **Structure:** Examine data types of features (e.g., numeric, categorical) and identify any missing values.
Univariate Analysis	• **Mean:** Calculate the average value of each feature (e.g., TSH levels). • **Median:** Determine the middle value of each feature. • **Mode:** Find the most frequently occurring value for categorical features (e.g., class labels like benign/malignant). • **Distribution:** Use histograms or density plots to visualize the distribution of each feature. For instance, visualize the distribution of TSH levels for different thyroid classifications.
Bivariate Analysis	• **Correlation:** Compute the correlation matrix to see how features relate to each other (e.g., TSH vs. T4 levels). Use Pearson, Spearman, or Kendall correlation coefficients depending on feature types.

	• **Scatter Plots:** Plot scatter plots to visualize relationships between pairs of features. For example, plot TSH vs. Free T4 to check for any apparent trends or clusters.
Multivariate Analysis	• **Patterns and Relationships:** Explore interactions between multiple features. Techniques like Principal Component Analysis (PCA) can help visualize the data in lower dimensions and identify patterns. • **Pair Plots:** Use pair plots to visualize relationships and distributions among features in the dataset. • **Heatmaps:** Use heatmaps of the correlation matrix to see the relationships between features.
Outliers and Anomalies	• **Identification:** Detect outliers using statistical methods like z-scores, or visualization tools like box plots. • **Treatment:** Depending on the impact, you may choose to remove outliers, transform them, or keep them. For instance, if an outlier in TSH levels is significantly different from the norm and doesn't fit with the general trend, it might be worth investigating further.
Data Preprocessing Code Screenshots	

Loading Data	import pandas as pd
	file_path = 'path/to/your/dataset.csv' # Replace with the actual file path
	df = pd.read_csv(file_path)
	
	# Display the first few rows of the dataset
	Print
	(df.head())

Handling Missing Data	import pandas as pd
	# Load the dataset
	file_path = 'path/to/your/dataset.csv' # Replace with the actual file path
	df = pd.read_csv(file_path)
	
	# Display the first few rows of the dataset
	print("Initial Dataset:\n", df.head())
	
	# Identify missing values
	missing_values = df.isnull().sum()

	print("\nMissing Values:\n", missing_values)
	
	# Handle missing values
	# Option 1: Drop rows with missing values
	df_dropped = df.dropna()
	print("\nDataset after dropping rows with missing values:\n", df_dropped.head())
	
	# Option 2: Fill missing values with a specified value (e.g., mean of the column)
	df_filled = df.fillna(df.mean())
	print("\nDataset after filling missing values with column mean:\n", df_filled.head())
	
	# Option 3: Fill missing values using forward fill method
	df_ffill = df.fillna(method='ffill')
	print("\nDataset after forward fill:\n", df_ffill.head())
	
	# Option 4: Fill missing values using backward fill method
	df_bfill = df.fillna(method='bfill')
	print("\nDataset after backward fill:\n", df_bfill.head())

Data Transformation	import pandas as pd
	from sklearn.preprocessing import StandardScaler, MinMaxScaler
	
	# Load the dataset
	file_path = 'path/to/your/dataset.csv' # Replace with the actual file path
	df = pd.read_csv(file_path)
	
	# Display the first few rows of the dataset
	print("Initial Dataset:\n", df.head())
	
	# Select columns to be scaled/normalized
	columns_to_transform = ['column1', 'column2', 'column3'] # Replace with actual column names
	
	# Standard Scaling (z-score normalization)
	scaler = StandardScaler()
	df_scaled = df.copy()
	df_scaled[columns_to_transform] = scaler.fit_transform(df_scaled[columns_to_transform])
	print("\nDataset after Standard Scaling:\n", df_scaled.head())
	
	# Min-Max Scaling

	min_max_scaler = MinMaxScaler() df_minmax_scaled = df.copy() df_minmax_scaled[columns_to_transform] = min_max_scaler.fit_transform(df_minmax_scaled[columns_to_transform]) print("\nDataset after Min-Max Scaling:\n", df_minmax_scaled.head())

Feature Engineering	import pandas as pd
	
	# Load the dataset
	file_path = 'path/to/your/dataset.csv' # Replace with the actual file path
	df = pd.read_csv(file_path)
	
	# Display the first few rows of the dataset
	print("Initial Dataset:\n", df.head())
	
	# Create new features or modify existing ones
	
	# Example 1: Creating a new feature based on existing columns
	# Assuming 'T3' and 'T4' are existing columns
	df['T3_T4_ratio'] = df['T3'] / df['T4']
	
	# Example 2: Creating a binary feature
	# Assuming 'Age' is an existing column, create a binary feature for age group
	df['is_senior'] = df['Age'].apply(lambda x: 1 if x >= 65 else 0)
	
	# Example 3: Binning a continuous variable
	# Assuming 'TSH' is an existing column
	bins = [0, 0.4, 4.0, float('inf')]
	labels = ['Low', 'Normal', 'High']
	df['TSH_level'] = pd.cut(df['TSH'], bins=bins, labels=labels, right=False)
	
	# Example 4: Encoding categorical variables
	# Assuming 'Gender' is a categorical column
	df = pd.get_dummies(df, columns=['Gender'], drop_first=True)
	
	# Display the first few rows of the modified dataset
	print("\nModified Dataset:\n", df.head())
	
	# Save the modified dataset to a new CSV file
	modified_file_path = 'path/to/your/modified_dataset.csv' # Replace with the desired file path

	df.to_csv(modified_file_path, index=False) print(f"Modified dataset saved to {modified_file_path}")

Save Processed Data	import pandas as pd
	
	# Load the dataset
	file_path = 'path/to/your/dataset.csv' # Replace with the actual file path
	df = pd.read_csv(file_path)
	
	# Example of data processing (handling missing values and scaling)
	# Handle missing values (fill with column mean)
	df_filled = df.fillna(df.mean())
	
	# Scaling the data (Standard Scaling)
	from sklearn.preprocessing import StandardScaler
	columns_to_transform = ['column1', 'column2', 'column3'] # Replace with actual column names
	scaler = StandardScaler()
	df_scaled = df_filled.copy()
	df_scaled[columns_to_transform] = scaler.fit_transform(df_scaled[columns_to_transform])
	
	# Save the cleaned and processed dataset to a new CSV file
	processed_file_path = 'path/to/your/processed_dataset.csv' # Replace with the desired file path
	df_scaled.to_csv(processed_file_path, index=False)
	
	print(f"Processed dataset saved to {processed_file_path}")