DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS89043
PROJECT NAME	Rising Waters: A Machine Learning Approach to Flood Prediction
MAXIMUM MARKS	3 MARKS

6.3 DATA PREPROCESSING:

Handling Missing Values

- Sometimes you may find some data missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values for continuous and for categorical we make use of mode values and replace the missing data.
- Word "True" that the particular column has missing values, we can also see the count of missing values in each column by using isnull().sum function.

```
#checking null values
dataset.isnull().any()
```

- isnull()

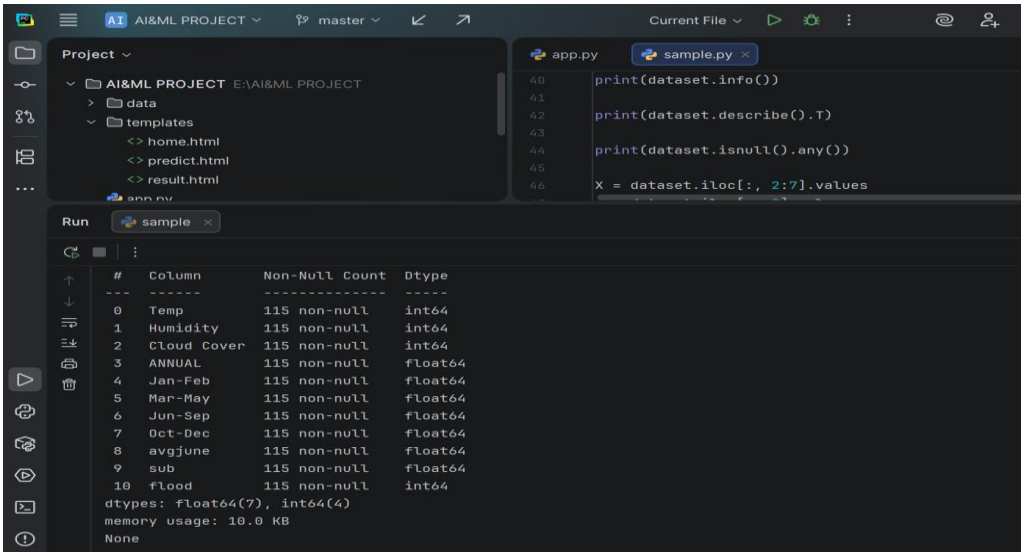
- It checks each cell in the dataset.
- If a value is missing, it returns True.
- If a value is present, it returns False.

- sum()

- It counts the number of True values in each column.
- Since True = 1 and False = 0,
- It gives the total number of missing values in each column.

- What You Understand From This

- Which columns have missing data
- How many values are missing
- Whether you should drop or fill them



Handling outliers:

With the help of boxplot, outliers are visualized (refer activity 3 univariate analysis). And here we are going to find upper bound and lower bound of Na to K feature with some mathematical formula.

- To find the upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.
- If outliers are removed, we lose more data. It will impact model performance.
- Here removing outliers is impossible. So, the capping technique is used on outliers.
- Capping: Replacing the outliers with upper bound values.

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project, we are using feature mapping and label encoding.

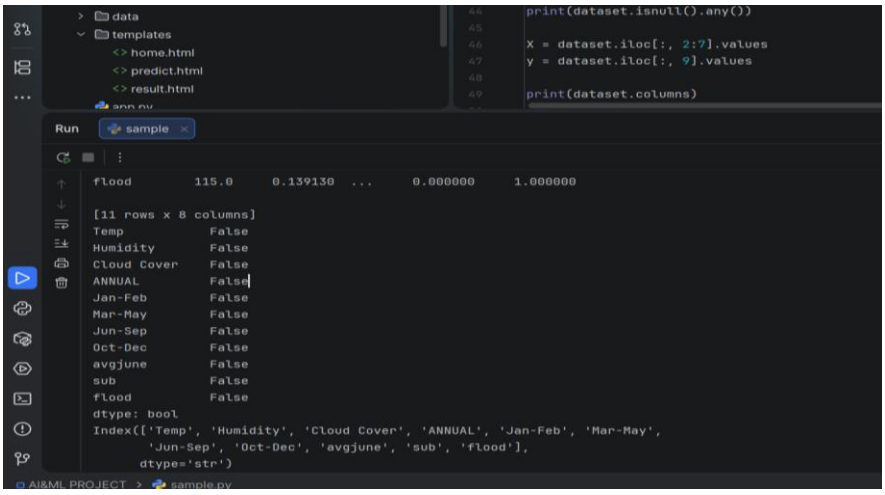
Splitting the Dataset into Dependent and Independent variables.

- In machine learning, the concept of the dependent variable (y) and independent variables(x) is important to understand. Here, the Dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset. We can denote with any symbol (alphabets). In our dataset, we can say that class is the dependent variable and all other columns are independent. But in order to select the independent columns, we will be selecting only those columns which are highly correlated and some value to our dependent column.
- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.
- Let's create out independent and dependent variables:

```
#independent features
x=dataset.iloc[:,2:7].values

#dependent feature
y=dataset.iloc[:,9:].values
```

- In the above code we are creating a Data Frame of the independent variable x with our selected columns and for the dependent variable y, we are only taking the class column.
- Where Data Frame is used to represent a table of data with rows and columns.



Split the dataset into Train set and Test set

- ·Now split our dataset into a train set and test using train_test_split class from sci-kit learn library.
- **Train_test_split:** used for splitting data arrays into training data and for testing data.

```
#split the data into train and test set from our x and y
#import train_test_split fucntion
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,random_state=10)
```

Feature scaling:

- Standard Scaler: Sklearn its main scaler, the StandardScaler, uses a strict definition of standardization to standardize data. It purely centers the data by using the following formula, where u is the mean and s is the standard deviation.

```
#import Standardscaler
from sklearn.preprocessing import StandardScaler
#create object to Standardscaler class
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

- Save the StandardScaler

```
#import dump class from joblib
from joblib import dump
dump(sc,"transform.save")
```