National University of Computer & Emerging Sciences, Karachi

FALL 2024, LAB MANUAL – 02

DATA WRANGLING

COURSE CODE :	**AL3002**
INSTRUCTOR :	**Usama Bin Umar**

OBJECTIVE

1. Importance of Data Preprocessing in ML
2. How to Combine Dataset and purpose of Groupby Function
3. How to handle Missing record
4. How to handle Categorical Features
5. How to handle Imbalance data
6. How to perform Feature Selection
7. How to perform Feature Scaling

HOW TO COMBINE DATASET:

Data collection is the first and crucial step of ML cycle. Data can be collected from various means that we have discussed earlier. The data can be collected in a group and then combine to perform a single study. There are several ways to combine dataset in Pandas. These methods are Merge, Join, Concat and append etc.

Combining Two DataFrames using Append

```
newdata = data1.append(data2,ignore_index=True)
newdata
```

Combining Two DataFrames using Concat

```
pd.concat([data1,data2],ignore_index=True,axis=0)
```

Combining Two DataFrames using Merge

```
mergeddata=data1.merge(data2,how='inner',on='age')
```

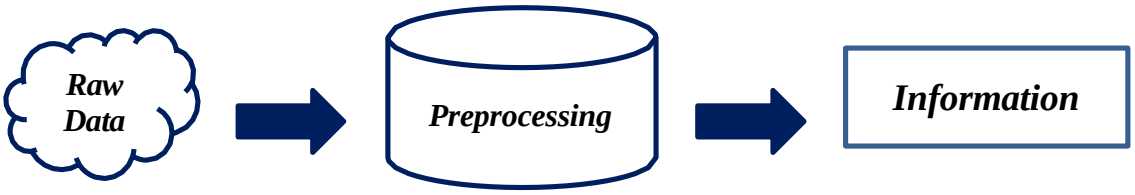
Join and merge combine data from multiple DataFrames based on specified conditions.
Concat and append add data to existing DataFrames (either vertically or horizontally) without any merging. Merge provides more flexibility for different types of joins and customization options. Concat and append are simpler and focused on straightforward concatenation.

OPEN SOURCE REPOSITORIES

There are numerous open-source repositories where you can find datasets for various tasks. Some of them are Kaggle, UCI machine learning repository, CVPR dataset, data world, OpenAI dataset, Microsoft research open dataset, ImageNet, COCO, Common crawl etc.

DATA PREPROCESSING

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for a machine learning model.



After performing Exploratory Data Analysis, the next step is to preprocess the data to extract useful information required to train a Machine Learning model.

DATA CLEANING

Data cleaning, also known as data cleansing, is a crucial step in the data preprocessing process. It involves identifying and correcting errors, inconsistencies, and inaccuracies in raw data to ensure that the data is accurate, reliable, and suitable for analysis or modeling. Data cleaning helps improve the quality of the dataset and prevents these issues from affecting the results of subsequent analyses or machine learning models.

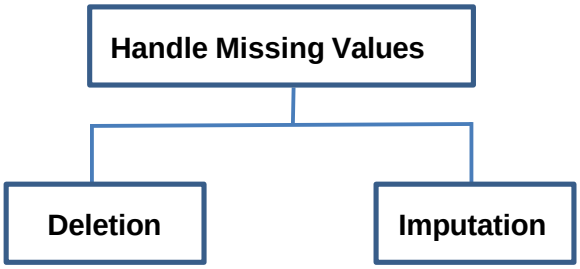
METHODS OF DATA CLEANING

1. Hanlde Missing Values
2. Removing Duplicates
3. Handling Outliers

HOW TO HANDLE MISSING RECORDS

Missing records in machine learning refer to instances in a dataset where one or more attributes lack data. Treating missing values is crucial because it maintains data integrity, enhances model performance, ensures unbiased analysis, and prevents skewed results or predictions

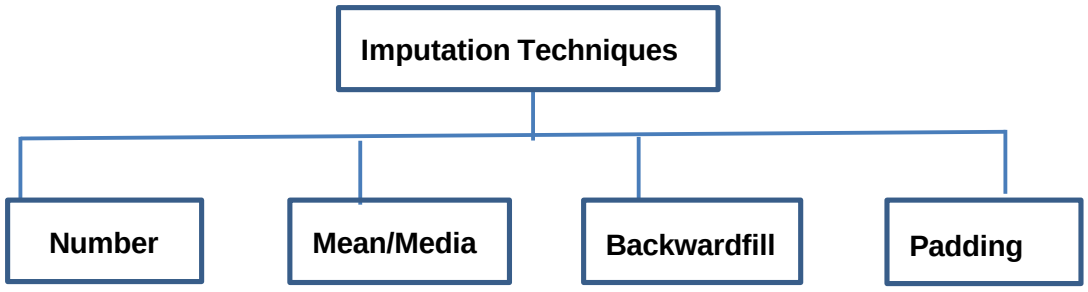
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1.0	0.0	125.0	212.0	0.0	1.0	168.0	0.0	1.0	2.0	2.0	3.0	0.0
1	53	1.0	0.0	140.0	203.0	1.0	0.0	155.0	1.0	3.1	0.0	0.0	3.0	0.0
2	70	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	2.6	0.0	0.0	3.0	0.0
3	61	1.0	0.0	148.0	203.0	0.0	1.0	161.0	NaN	NaN	NaN	NaN	NaN	NaN
4	62	0.0	0.0	138.0	294.0	1.0	1.0	106.0	0.0	1.9	1.0	3.0	2.0	0.0



Deletion: In some cases, you might opt to delete instances with missing values. However, this approach can lead to loss of valuable data, especially if the missing data is widespread.

Drop Records that Contain Missing Values	
`df.dropna(inplace=True)`	
Shape Before Drop	**Shape After Drop**
(1025, 14)	(1014, 14)

Imputation: This involves filling in missing values with estimated or imputed values. Imputation methods can be simple, such as filling with the mean, median, or mode of the available values, or more complex, like using regression models to predict missing values based on other features.



Fill Records Using Number

```
#fill by constant
df.fillna( 88 , inplace=True )
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1.0	0.0	125.0	212.0	0.0	1.0	168.0	0.0	1.0	2.0	2.0	3.0	0.0
1	53	1.0	0.0	140.0	203.0	1.0	0.0	155.0	1.0	3.1	0.0	0.0	3.0	0.0
2	70	88.0	88.0	88.0	88.0	88.0	88.0	88.0	1.0	2.6	0.0	0.0	3.0	0.0
3	61	1.0	0.0	148.0	203.0	0.0	1.0	161.0	88.0	88.0	88.0	88.0	88.0	88.0
4	62	0.0	0.0	138.0	294.0	1.0	1.0	106.0	0.0	1.9	1.0	3.0	2.0	0.0

Fill Records Using Mean, Media or Mode

```
#fill record using Mean
df.fillna(df.mean(),inplace=True)
df.head()
```

Fill Record using Median

```
#fill record using Median
df.fillna(df.median(),inplace=True)
```

Fill Record using Mode

```
#fill record using Mode
df.fillna(df.mode(),inplace=True)
```

Fill Record using Padding

```
#fill record using Padding
df.fillna(df.pad(),inplace=True)
```

Fill Record using Backwardfill

```
#fill record using Padding
df.fillna(df.backfill(),inplace=True)
```

Data can easily be imputer using a constant number, mean, median and mode. Padding and Backfill. Backward fill imputation allows you to perform backward fill imputation on your Dataframe. It fills missing values with the next available value from the same column whereas padding fill the values with the most recent available value from the same column.

HOW TO HANDLE CATEGORICAL VALUES

Categorical features are distinct data attributes that represent different groups or labels. They can be nominal (with no inherent order) or ordinal (with a specific order). For effective use in machine learning, categorical features are transformed into numerical values through different techniques. There are two major types of encoding techniques
One hot Encoding
Label Encoding.

Checking Categorical Features

```
df.select_dtypes(include=['object']).dtypes
```

Label Encoding

```
# Implementing Label Encoding Using Pandas
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['GENDER']=le.fit_transform(df['GENDER'])
print('============= after encoding ======')
print(df)
```

One Hot Encoding

```
# get dummies method
target=pd.get_dummies(df['LUNG_CANCER'])
print(target)
datanew=pd.concat([df,target],axis=1)
print(datanew)
```

One-Hot Encoding converts each category into a separate binary column. Suitable for nominal (unordered) categorical features, creates a binary column for each category, indicating presence (1) or absence (0) and Prevents false ordinal relationships, but can lead to high dimensionality.
Label Encoding assigns a unique integer to each category, Suitable for ordinal (ordered) categorical

features, converts categories to numerical values based on order or assigned integers and retains order information but may introduce unintended ordinal relationships.

HOW TO HANDLE IMBALANCE DATA

Handling imbalanced data is crucial to prevent bias in machine learning models and to ensure that the model performs well across all classes, especially when the classes are not equally represented in the dataset.

Checking Balance of Data

```
#Numerically Analyze
df['LUNG_CANCER'].value_counts()

#Graphically Analyze
import seaborn as sns
sns.countplot(x='LUNG_CANCER',data=df,palette='RdBu_r')
```

SMOTE (Over Sampling)

```
#Split Target Variable
y=df.pop('LUNG_CANCER')

from imblearn.combine import SMOTETomek
# Implementing Oversampling for Handling Imbalanced
smk = SMOTETomek(random_state=42)
xdata,ydata=smk.fit_resample(df,y)
xdata.shape,ydata.shape
```

```
from collections import Counter
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(ydata)))
```

```
Original dataset shape Counter({'YES': 270, 'NO': 39})
Resampled dataset shape Counter({'YES': 265, 'NO': 265})
```

NEAR MISS (Under Sampling)

```
# UnderSampling using NearMiss
from imblearn.under_sampling import NearMiss
nm = NearMiss()
X_res,y_res=nm.fit_resample(df,y)
X_res.shape,y_res.shape
```

```
from collections import Counter
print('Original dataset shape {}'.format(Counter(y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))
```

```
Original dataset shape Counter({'YES': 270, 'NO': 39})
Resampled dataset shape Counter({'NO': 39, 'YES': 39})
```

In Oversampling we increase the number of instances in the minority class by duplicating existing instances or generating synthetic samples (e.g., using techniques like SMOTE - Synthetic Minority Over-sampling Technique). In Under-sampling we try to decrease the number of instances in the majority class by randomly removing instances.

FEATURE SELECTION

Feature selection in machine learning is the process of selecting a subset of relevant features (input variables or attributes) from the original set of features to use in building a predictive model. The goal of feature selection is to improve model performance by reducing the dimensionality of the dataset while retaining the most informative and significant features.

There are 4 different categories of Feature Selection

1. Filter methods
2. Wrapper methods
3. Embedded methods
4. Hybrid methods

FILTER METHOD: In this method, features are dropped based on their relation to the output, or how they are correlating to the output.
We use **correlation** to check if the features are positively or negatively correlated to the output labels and drop features accordingly. E.g.: Information Gain, Chi-Square Test, Fisher's Score, etc.

WRAPPER METHOD: We split our data into subsets and train a model using this.
Based on the output of the model, we add and subtract features and train the model again. It forms the subsets using a greedy approach and evaluates the accuracy of all the possible combinations of features. E.g.: Forward Selection, Backwards Elimination, etc.

INTRINSIC METHOD: This method combines the qualities of both the Filter and Wrapper method to create the best subset.

FILTER METHOD	WRAPPER METHOD	INTRINSIC METHOD
1. Information Gain 2. Chi Square Test 3. Fisher's score 4. Correlation Coefficient 5. Variance Threshold 6. Mean absolute difference 7. Dispersion Ratio	1. Forward Feature Selection 2. Backward Selection 3. Exhaustive FS 4. Recursive FS	1. Lasso Regularization (l1) 2. Random Forest Importance 3. Ridge(L2) 4. Elastic (l1 n l2) 5. Algorithm Based Approach

Feature Selection using VARIANCE THRESHOLD

```
### It will remove zero variance features
from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=0)
var_thres.fit_transform(data)

#Checking Variance of Variables
var_thres.variances_
#Checking Which Variance is above or below the Threshold
var_thres.get_support()
data.columns[var_thres.get_support()]

lowVarFeatures = [column for column in data.columns
                    if column not in data.columns[var_thres.get_support()]]

#Dropping Constant Features
data.drop(lowVarFeatures,axis=1,inplace=True)
```

Feature Selection using PEARSON CORRELATION

```
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                colname = corr_matrix.columns[i]  # getting the name of column
                col_corr.add(colname)
    return col_corr

corr_features = correlation(df, 0.4)
len(set(corr_features))
df.drop(corr_features,axis=1)
```

FEATURE SCALING

Feature scaling in machine learning is the process of adjusting the magnitudes of numerical features to a common scale. This helps algorithms work better with features of varying scales. Two common methods are:

1. **Standardization:** Scaling features to have a mean of 0 and a standard deviation of 1.
2. **Min-Max Scaling:** Scaling features to a specific range, often between 0 and 1.

Scaling is crucial for distance-based algorithms and optimization techniques. It ensures fair treatment of features and can improve model performance. Apply scaling after splitting data into training and testing sets to prevent data leakage.

Feature Scaling using MinMax Algorithm

```
from sklearn.preprocessing import MinMaxScaler #normalization
mc = MinMaxScaler()
X_train = mc.fit_transform(x_train)
X_test = mc.transform(x_test)
print(X_train)
```

Feature Scaling using Standard Algorithm

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
print(X_train)
```

LAB TASKS

Task # 1

A. Download the dataset and explore how to merge these dataset
B. Combine dataset **Lab2 D1A** with **Lab2 D1B** in such a way that it doesn't contain any duplicate column. The resultant dataset consist of these columns and the final shape will be

```
Index(['fid', 'name', 'population', 'county', 'latitude', 'longitude', 'level',
       'enrollment', 'level.1'],
      dtype='object')
Final Records :  (27032, 9)
```

C. Combine dataset **Lab2 D1A** with **Lab2 D1C** using merge method to extract similar records in a new Dataframe **"comboAC"** having 4221333 records

```
(4221333, 7)
Index(['fid', 'name', 'population', 'county', 'latitude', 'city', 'score'], dtype='object')
```

Task # 2

A. Customized you own dataset with the name "customizedData", add at least one attribute that should be similar to **Lab2 D1A, Lab2 D1B, Lab2 D1C** dataset , now add 3 attributes of Size (small, ,medium, and high), cardinal direction (North, South, East and West) , Timings (full time , part time) and add 2 attributes of your own choice, one attribute should be categorical and one should be continuous.

B. Merge "customizedData" with Lab2 D1A, Lab2 D1B, Lab2 D1C and produce a resultand dataset with the name of "modifiedData" and explore/ analyze its number of records and features before and after merging with the technique of similar records joining.

Task # 3

How to organize your code (Create a Text block for Importing Libraries in next cell import required library now Create a Heading for Data Preprocessing and the add a new cell for Data analysis. Create headings for each cell)

A. Download the dataset from Here
B. Import your dataset in Colab or Jupyter notebook
C. Calculate the correlation between these variables,Var3,Var38,Var15, imp_op_var39_comer_ult1
D. Check whether the data is linear or not and write a brief explanation what you have analyzed in text cell
E. Check whether the data contain any missing record, if yes then perform imputation using an average method.

F. In your dataset, you have some interesting variables. Think of multi-variable research questions that you can explore with these data and explore. You need to do at least 5 explorations that include data visualizations, numerical summary.
G. Find out the unique category in target variable and check whether your dataset is balanced or not.
H. If dataset is not balanced, then handle your dataset and balance it using Up sampling
I. Find out the total number of features and records and perform feature selection using Pearson Correlation having threshold equal to 65%.
J. Make a copy of your dataset and perform feature selection other than Pearson and Variance threshold.

Task # 4

A. Create a survey form (ask for the approval in order to avoid duplicate content) , ask some questions , Make sure to choose your question wisely, your attributes should reflect more to your problem statement. User have to answer atleast 5 questions and the remaining one will be depend on user, whether to answer or not.
B. The questionnaire will contains at least 10 questions and you have to collect dataset from minimum 100 individuals
C. After collecting dataset, Perform some Statistical analysis over it, Do some Graphical Visualization, Check whether the dataset you have collected is balanced or not.
D. Perform Data wrangling, If the dataset contain any missing records try to handle these missing values wisely.
E. If you found your dataset is not balanced then choose a technique other than smote or NearMiss
F. Perform feature selection technique other than Variance Threshold and Pearson correlation and explain in a text cell its working.
G. If the dataset contain any categorical feature then encode it using Dummy Encoding, and explain the difference between dummy encoding and one hot encoding
H. Check whether your dataset contain any duplicate records, if it does, then handle these records with atleast 2 techniques.