Experiment 01 - Data Preparation

|  |  |
| --- | --- |
| Roll No. |  |
| Name |  |
| Class | D15-A |
| Subject | DS using Python Lab |
| LO Mapped | LO1: Understand the concept of Data science process and associated terminologies to solve real-world problems |
|  |  |

**Aim**:

To perform data preprocessing on the dataset using python.

**Introduction**:

**Data preprocessing**

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithms.

**Why is Data Preprocessing important?**

Preprocessing of data is mainly to check the data quality. The quality can be checked by the following:

* Accuracy: To check whether the data entered is correct or not.
* Completeness: To check whether the data is available or not recorded.
* Consistency: To check whether the same data is kept in all the places that do or do not match.
* Timeliness: The data should be updated correctly.
* Believability: The data should be trustable.
* Interpretability: The understandability of the data.

**Major Tasks in Data Preprocessing:**

1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation

**Dataset Attributes and its Types:**

Dataset:   
Annual Ticket sales of Hollywood movies. (<https://www.kaggle.com/johnharshith/hollywood-theatrical-market-synopsis-1995-to-2021>)

Shape:   
11914 rows, 16 columns

Columns and datatypes:

|  |  |  |
| --- | --- | --- |
| **Attribute/ Column Name** | **Data Type** | **Description** |
| Year | Date Time | Year of collection |
| Tickets Sold | Long | Number of tickets sold |
| Total Box Office Collection | Long | Total collection of movie tickets |
| TOTAL INFLATION ADJUSTED BOX OFFICE | Long | Inflation adjusted collection |
| AVERAGE TICKET PRICE | float64 | Average price of tickets sold |

**Indexing - add index field to the dataset**

Indexing in pandas means simply selecting particular rows and columns of data from a DataFrame. Indexing could mean selecting all the rows and some of the columns, some of the rows and all of the columns, or some of each of the rows and columns. Indexing can also be known as Subset Selection.

There are a lot of ways to pull the elements, rows, and columns from a DataFrame. There are some indexing methods in Pandas that help in getting an element from a DataFrame. These indexing methods appear very similar but behave very differently. Pandas support four types of Multi-axes indexing they are -

* Dataframe.[ ] ; This function also known as indexing operator
* Dataframe.loc[ ] : This function is used for labels.
* Dataframe.iloc[ ] : This function is used for positions or integer-based
* Dataframe.ix[] : This function is used for both label and integer-based

DataFrame has a set\_index() method which takes a column name (for a regular Index) or a list of column names (for a MultiIndex).

In our dataset, we used this method for adding an index field/column to the dataset.

**import pandas as pd**

**d = pd.read\_csv('AnnualTicketSales.csv')**

**d.reset\_index(level=0, inplace=True)**

**print(d)**

**Data Cleaning - removing missing values**

**Data Cleaning**

Data Cleaning means the process of identifying the incorrect, incomplete, inaccurate, irrelevant or missing part of the data and then modifying, replacing or deleting them according to the necessity. Data cleaning is considered a foundational element of basic data science.

Machine Learning is a data-driven AI. In machine learning, if the data is irrelevant or error-prone then it leads to an incorrect model building. As much as you make your data clean, as much as you can make a better model. So, we need to process or clean the data before using it. Without the quality data,it would be foolish to expect anything good.

**Missing Values/Data**

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset. In the dataset, blank shows the missing values. In Pandas, usually, missing values are represented by NaN.

There can be multiple reasons why certain values are missing from the data.

Reasons for the missing data from the dataset affect the approach of handling missing data. So it’s necessary to understand why the data could be missing.

Some of the reasons are listed below:

* Past data might get corrupted due to improper maintenance.
* Observations are not recorded for certain fields due to some reasons. There might be a failure in recording the values due to human error.
* The user has not provided the values intentionally.

**Types of Missing Data**

Missing data is grouped into three broad categories:

1. Missing completely at random (MCAR)

Data is missing completely at random if all observations have the same likelihood of being missing.

1. Missing at random (MAR)

When data is missing at random (MAR) the likelihood that a data point is missing is not related to the missing data but may be related to other observed data.

1. Missing not at random (MNAR)

When data is missing not at random (MNAR) the likelihood of a missing observation is related to its values. It can be difficult to identify MNAR data because the values of missing data are unobserved. This can result in distorted data.

**Handling Missing Data**

Handling missing values is an important step in data cleaning that can impact model validity and reliability. It is important to handle the missing values appropriately.

* Many machine learning algorithms fail if the dataset contains missing values. However, algorithms like K-nearest and Naive Bayes support data with missing values.
* You may end up building a biased machine learning model which will lead to incorrect results if the missing values are not handled properly.
* Missing data can lead to a lack of precision in the statistical analysis.

Checking for missing values: The first step in handling missing values is to look at the data carefully and find out all the missing values. The following code shows the total number of missing values in each column:

df.isna().sum()

Analyze each column with missing values carefully to understand the reasons behind the missing values as it is crucial to find out the strategy for handling the missing values.

There are 2 primary ways of handling missing values:

1. Deleting the Missing Values
2. Imputing the Missing Values

**Deleting the Missing Value**

Generally, this approach is not recommended. It is one of the quick and dirty techniques one can use to deal with missing values.

If the missing value is of the type Missing Not At Random (MNAR), then it should not be deleted. If the missing value is of type Missing At Random (MAR) or Missing Completely At Random (MCAR) then it can be deleted.

The disadvantage of this method is one might end up deleting some useful data from the dataset.

There are 2 ways one can delete the missing values:

1. Deleting the entire row

If a row has many missing values then you can choose to drop the entire row.

If every row has some (column) value missing then you might end up deleting the whole data.

Code:

df = df.dropna(axis=0)

2. Deleting the entire column

If a certain column has many missing values then you can choose to drop the entire column.

Code:

df = df.drop(['Dependents'],axis=1)

**Imputing the Missing Value**

There are different ways of replacing the missing values:

1. Replacing with a arbitrary/default value

If you can make an educated guess about the missing value then you can replace it with some arbitrary value using the following code.

Code:

df['Dependents'] = df['Dependents'].fillna(0)

2. Replacing With Mean Value

This is the most common method of imputing missing values of numeric columns. If there are outliers then the mean will not be appropriate. In such cases, outliers need to be treated first.

Code:

df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())

3. Replacing With Mode

Mode is the most frequently occurring value. It is used in the case of categorical features.

Code:

df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

4. Replacing with previous value – Forward fill

In some cases, imputing the values with the previous value instead of mean, mode or median is more appropriate. This is called forward fill. It is mostly used in time series data

You can use ‘fillna’ function with the parameter ‘method = ffill’

Code:

df.fillna(method=‘ffill')

5. Replacing with next value – Backward fill

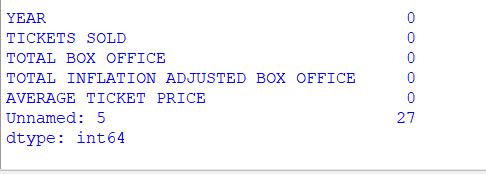
In backward fill, the missing value is imputed using the next value

Code:

test.fillna(method=‘bfill')

**Missing data in our Hollywood ticket sales dataset**

**dataset:**



**Data Cleaning - removing noisy values**

*(explain and execute how noisy values can be identified and then how they can be handled*

*List the attributes which have outliers and print top 10 outliers)*

**What are Outliers?**

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. In a sense, this definition leaves it up to the analyst (or a consensus process) to decide what will be considered abnormal. Before abnormal observations can be singled out, it is necessary to characterize normal observations.

**Data Transformation**

Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general. It is an imperative step in feature engineering that facilitates discovering insights. This article will cover techniques of numeric data transformation: log transformation, clipping methods, and data scaling.

Why need data transformation?

* the algorithm is more likely to be biased when the data distribution is skewed
* transforming data into the same scale allows the algorithm to compare the relative relationship between data points better

When to apply data transformation?

When implementing supervised algorithms, training data and testing data need to be transformed in the same way. This is usually achieved by feeding the training dataset to building the data transformation algorithm and then apply that algorithm to the test set.

**Conversion of categorical into numeric data**:

In many Machine-learning or Data Science activities, the data set might contain text or categorical values (basically non-numerical values). Over your learning curve in AI and Machine Learning, one thing you would notice is that most of the algorithms work better with numerical inputs. Therefore, the main challenge faced by an analyst is to convert text/categorical data into numerical data and still make an algorithm/model to make sense out of it. Neural networks, which is a base of deep-learning, expects input values to be numerical.

There are many ways to convert categorical values into numerical values. Each approach has its own trade-offs and impact on the feature set. We would focus on using One-Hot Encoding and Label-Encoder. Both of these encoders are part of SciKit-learn library (one of the most widely used Python libraries) and are used to convert text or categorical data into numerical data which the model expects and performs better with.

**Label Encoding**

This approach is very simple and it involves converting each value in a column to a number. Consider a dataset of bridges having a column named bridge-types having below values. Though there will be many more columns in the dataset, to understand label-encoding, we will focus on one categorical column only.

preprocessing.LabelEncoder() of sklearn is used for label encoding.

In our dataset we use this on Vehicle Style and Model

**One-Hot Encoder**

Though label encoding is straight, it has the disadvantage that the numeric values can be misinterpreted by algorithms as having some sort of hierarchy/order in them. This ordering issue is addressed in another common alternative approach called ‘One-Hot Encoding’. In this strategy, each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column. Let’s consider the previous example of bridge type and safety levels with one-hot encoding.

In our dataset we use this on Transmission Type and Vehicle Size

**Data Normalization**

Feature scaling refers to putting the values in the same range or same scale so that no variable is dominated by the other.

Numerical data in the dataset can have a varied range i.e. one parameter may lie between 1 to 10 for all records whereas another parameter can lie between 1000 to 5000. Though data is logically correct but after passing to a particular algorithm, the features with higher magnitude become key parameters for that algorithm.

To avoid such situations feature scaling is performed using some statistical techniques like Min-Max scaling & Mean normalization. This creates a common range for all the parameters and thus removes Algorithmic bias.

**Normalization** is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Normalization equation



Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

* When the value of X is the minimum value in the column, the numerator will be 0, and hence X’ is 0
* On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X’ is 1
* If the value of X is between the minimum and the maximum value, then the value of X’ is between 0 and 1

To normalize your data, you need to import the MinMaxScalar from the sklearn library and apply it to our dataset.

**Standardization** is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Standardization equation



To standardize your data, you need to import the StandardScalar from the sklearn library and apply it to our dataset.

**Conclusion**:

Thus we have understood how to perform data preprocessing which can further be taken into exploratory data analysis and further in the Model preparation sequence.