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:   
Car Features and MSRP ([Car Features and MSRP | Kaggle](https://www.kaggle.com/CooperUnion/cardataset))

Shape:   
11914 rows, 16 columns

Columns and datatypes:

| **Attribute/ Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| Make | object | Car Brand Name |
| Model | object | Car Model Name |
| Year | int64 | Year of Model Release |
| Engine Fuel Type | object | Type of Fuel |
| Engine HP | float64 | Engine Horsepower |
| Engine Cylinders | float64 | Number of cylinders in the engine |
| Transmission Type | object | Transmission Type (Auto or Manual) |
| Driven Wheels | object | Driven Wheels Specification |
| Number of Doors | float64 | Door Count |
| Market Category | object | Type(s) of different car categories the vehicle fits into |
| Vehicle Size | object | General Idea of Vehicle Size |
| Vehicle Style | object | General Idea of Vehicle Style |
| Highway MPG | int64 | Average MPG (miles per gallon) value on highways |
| City MPG | int64 | Average MPG (miles per gallon) value on cities |
| Popularity | int64 | Popularity Rating |
| MSRP | int64 | MSRP as of Vehicle Launch |

Numerical attributes of the dataset:

Year, Engine HP, Engine Cylinders, Number of Doors, Highway MPG, City MPG, Popularity, MSRP

Categorical attributes of the dataset:

Transmission Type, Number of Doors, Vehicle Size

**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.

df = source

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

df.head()

**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')

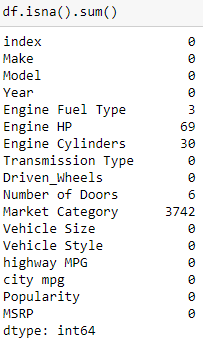
5. Replacing with next value – Backward fill

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

Code:

test.fillna(method=‘bfill')

**Missing data in our Car Features and MSRP dataset:**



**Data Cleaning - removing noisy values**

Noisy data is meaningless data. The term has often been used as a synonym for corrupt data. However, its meaning has expanded to include any data that cannot be understood and interpreted correctly by machines, such as unstructured text. Any data that has been received, stored, or changed in such a manner that it cannot be read or used by the program that originally created it can be described as noisy.

Noisy data unnecessarily increases the amount of storage space required and can also adversely affect the results of any data mining analysis. Statistical analysis can use information gleaned from historical data to weed out noisy data and facilitate data mining.

Noisy data can be caused by hardware failures, programming errors and gibberish input from speech or optical character recognition (OCR) programs. Spelling errors, industry abbreviations and slang can also impede machine reading.

**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.

**Reasons for outliers in data**

Most common causes of outliers on a data set:

1. Data Entry Errors: Human errors such as errors caused during data collection, recording, or entry can cause outliers in data.
2. Measurement Error (instrument errors): It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty.
3. Experimental errors (data extraction or experiment planning/executing errors)
4. Intentional (dummy outliers made to test detection methods)
5. Data processing errors (data manipulation or data set unintended mutations)
6. Sampling errors (extracting or mixing data from wrong or various sources)
7. Natural Outlier (not an error, novelties in data): When an outlier is not artificial (due to error), it is a natural outlier. Most real world data belong to this category.

**Problems caused by outliers**

1. Outliers in the data may cause problems during model fitting (esp. linear models).
2. Outliers may inflate the error metrics which give higher weights to large errors (example, mean squared error, RMSE).

**Types of Outliers**

1. Point or global Outliers: Observations anomalous with respect to the majority of observations in a feature. In-short A data point is considered a global outlier if its value is far outside the entirety of the data set in which it is found.

2. Contextual (Conditional) Outliers: Observations considered anomalous given a specific context. A data point is considered a contextual outlier if its value significantly deviates from the rest of the data points in the same context. Note that this means that same value may not be considered an outlier if it occurred in a different context. If we limit our discussion to time series data, the “context” is almost always temporal, because time series data are records of a specific quantity over time.

3. Collective Outliers: A collection of observations anomalous but appear close to one another because they all have a similar anomalous value.

**Methods to identify outliers in the data**

1. Box plots

Box plots are a visual method to identify outliers. Box plots are one of the many ways to visualize data distribution. Box plot plots the q1 (25th percentile), q2 (50th percentile or median) and q3 (75th percentile) of the data along with (q1–1.5\*(q3-q1)) and (q3+1.5\*(q3-q1)). Outliers, if any, are plotted as points above and below the plot.

2. IQR method

The IQR method is used by box plots to highlight outliers. IQR stands for interquartile range, which is the difference between q3 (75th percentile) and q1 (25th percentile). The IQR method computes lower bound and upper bound to identify outliers.

Lower Bound = q1–1.5\*IQR

Upper Bound = q3+1.5\*IQR

Any value below the lower bound and above the upper bound are considered to be outliers.

3. Z-score method

The Z-score method is another method for detecting outliers. This method is generally used when a variable's distribution looks close to Gaussian. Z-score is the number of standard deviations a value of a variable is away from the variable's mean.

Z-Score = (X-mean) / Standard deviation

when the values of a variable are converted to Z-scores, then the distribution of the variable is called standard normal distribution with mean=0 and standard deviation=1. The Z-score method requires a cut-off specified by the user, to identify outliers. The widely used lower end cut-off is -3 and the upper end cut-off is +3. The reason behind using these cut-offs is, 99.7% of the values lie between -3 and +3 in a standard normal distribution.

4. ‘Distance from the mean’ method (Multivariate method)

Unlike the previous methods, this method considers multiple variables in a data set to detect outliers. This method calculates the Euclidean distance of the data points from their mean and converts the distances into absolute z-scores. Any z-score greater than the pre-specified cut-off is considered to be an outlier.

**Handling outliers**

1. Deleting observations

Sometimes it’s best to completely remove those records from your dataset to stop them from skewing your analysis. We delete outlier values if it is due to data entry error, data processing error or outlier observations are very small in numbers. We can also use trimming at both ends to remove outliers. But deleting the observation is not a good idea when we have a small dataset.

2. Transforming values

Transforming variables can also eliminate outliers. These transformed values reduce the variation caused by extreme values.

1. Scaling
2. Log transformation
3. Cube Root Normalization
4. Box-transformation

These techniques convert values in the dataset to smaller values. If the data has too many extreme values or is skewed, this method helps to make your data normal. But These techniques do not always give you the best results. There is no loss of data from these methods. In all these methods, box cox transformation gives the best result.

3. Imputation

Like imputation of missing values, we can also impute outliers. We can use mean, median, zero value in these methods. Since we are imputing there is no loss of data. Here median is appropriate because it is not affected by outliers.

4. Separately treating

If there are a significant number of outliers and the dataset is small , we should treat them separately in the statistical model. One of the approaches is to treat both groups as two different groups and build individual models for both groups and then combine the output. But this technique is tedious when the dataset is large.

5. Winsorization

Winsorization is the process of replacing the extreme values of statistical data in order to limit the effect of the outliers on the calculations or the results obtained by using that data. The mean value calculated after such replacement of the extreme values is called winsorized mean.

6. Binning

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin. This has a smoothing effect on the input data and may also reduce the chances of overfitting in the case of small datasets

In our Car Features and MSRP dataset, we have outliers in:

1. Engine HP
2. Highway MPG
3. City MPG
4. Popularity

In our experiment we have removed outliers for the Engine HP attribute using the IQR method. The code for the same and top 10 outliers are:

# calculate the outlier cutoff based on Engine HP

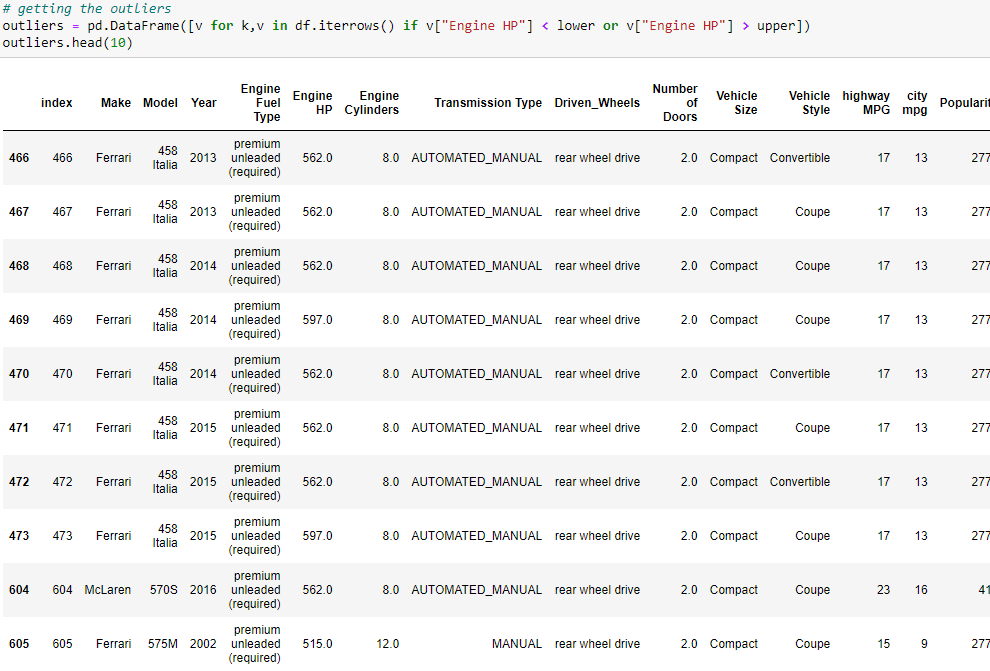
cut\_off = IQR["Engine HP"] \* 1.5

lower, upper = Q1["Engine HP"] - cut\_off, Q3["Engine HP"] + cut\_off

# getting the outliers

outliers = pd.DataFrame([v for k,v in df.iterrows() if v["Engine HP"] < lower or v["Engine HP"] > upper])

outliers.head(10)



**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.