DS JULY 2022 Batch
Module 12 – Exploratory Data Analysis

Topics

- Exploratory Data Analysis
- Missing values treatment
- Outlier detection and treatment
- Column Standardization
- Categorical variable treatment
- Feature Importance

Exploratory Data Analysis

Exploratory Data Analysis

Exploratory Data Analysis

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before starting our data analysis.

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

There are various steps involved in EDA which are

- Univariate analysis
- Bivariate analysis
- Missing values treatment
- Outlier analysis
- Column standardization
- Categorical features treatment.

Missing Values Treatment

Missing Values Treatment

Missing values

- Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset.
- The problem of missing value is quite common in many real-life datasets.
- Missing value can bias the results of the machine learning models and/or reduce the accuracy of the model.
- The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.
- In Pandas, usually, missing values are represented by **NaN.** It stands for **Not a Number.**

Reasons for data missing from the dataset

•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.

Checking Missing values in a dataset

Identifying Missing values

The first step in handling missing values is to look at the data carefully and find out all the missing values.

We can identify missing values in a dataset using following codes (Consider dataframe name is df).

Operator	Description	Syntax
isna	Check missing values counts in all columns of dataframe	df.isna().sum()
isnull	Check missing values counts in all columns of dataframe	df.isnull().sum()
notna	Check non-missing values counts in all columns of dataframe	df.notna().sum()
notnull	Check non-missing values counts in all columns of dataframe	df.notnull().sum()

Ways to handle the missing data

We can use following methods to deal with missing value.

1) Deleting Rows with missing values
2) Impute missing values for numeric variable
3) Impute missing values for categorical variable
4) Other Imputation Methods
5) Using algorithms which supports missing values
6) Prediction of missing values

Deleting Rows with missing values

Missing values can be handled by deleting the rows or columns having null values.

Deleting rows with missing values

If total number of rows with any missing values is less than 5% then we can directly remove all such rows.

Identifying percentage of missing values of columns

```
df.isna().mean()*100
```

Deleting all the rows with missing observations

```
df.dropna(how, subset, inplace)
```

how : 'any' - If any NA values are present, drop that row or column/
'all' - If all values are NA, drop that row or column.

subset : List of columns to be considered for Nan values

inplace : If True changes will be saved permanently

Deleting specific columns from dataframe

```
df.drop(columns, inplace)
```

columns : List of columns to be deleted

inplace : If True changes will be saved permanently

Impute missing values for numeric variables

- Columns in the dataset which are having numeric continuous values can be replaced with the mean, median, or mode of remaining values in the column.
- This method can prevent the loss of data compared to the earlier method.
- Replacing the above two approximations (mean, median) is a statistical approach to handle the missing values.
- When our variable is discrete or when the variable is skewed then we should use median/mode for filling values rather than mean.

Filling values with mean of the column

```
df[column].fillna(df[column].mean(),inplace=True)
```

Filling values with median of the column

```
df[column].fillna(df[column].median(),inplace=True)
```

Filling values with mode of the column

```
df[column].fillna(df[column].mode()[0],inplace=True)
```

Impute missing values for categorical variables

- When missing values is from categorical columns (string or numerical) then the missing values can be replaced with the most frequent category (mode)
- If the number of missing values is very large then it can be replaced with a new category.

Filling values with mode of the column

```
df[column].fillna(df[column].mode()[0],inplace=True)
```

Other imputation methods

- Depending on the nature of the data or data type, some other imputation methods may be more appropriate to impute missing values.
- For the data variable having longitudinal behavior, it might make sense to use the last valid observation to fill the missing value. This is known as the Last observation carried forward (LOCF) method.

Filling values with mode of the column

```
df[column].fillna(method, inplace=True)
```

method : Method to use for filling.

‘ffill’ - use previous valid observation to fill gap

‘bfill’ - use next valid observation to fill gap

Using algorithms which supports missing values

- All the machine learning algorithms don’t support missing values but some ML algorithms are robust to missing values in the dataset.
- There are many machine learning algorithms which support Nan values and internally treat them like XgBoost,Lightgbm and catboost which works on data with missing values as well.

Prediction of missing values

- Many a times we can also use machine learning algorithms to predict missing values.
- All those rows for which values for variable is not missing will be considered as training data.
- All those rows for which values for variable is missing will be considered as test data.
- If the variable is categorical we can use **classification algorithm** and if variable is continuous we will use **regression algorithm**.

Outlier detection and Treatment

Outliers

What are outliers?

- Outliers are those data points that are significantly different from the rest of the dataset.
- They are often abnormal observations that skew the data distribution, and arise due to inconsistent data entry, or erroneous observations.

Reasons for presence of outliers in data

- Errors during data entry or a faulty measuring device (a faulty sensor may result in extreme readings).
- Natural occurrence (Like Covid 19)

Why should we detect outliers?

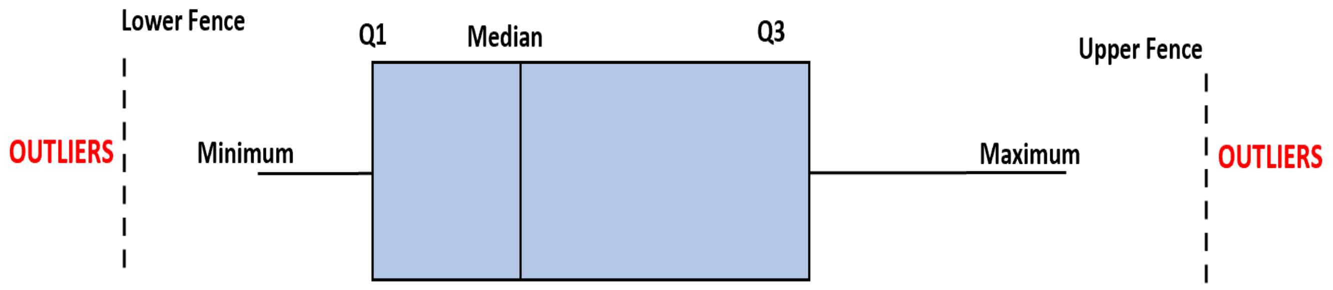
- Outliers in the data may causes problems during model fitting (esp. linear models).
- Outliers may inflate the error metrics which give higher weights to large errors (example, mean squared error, RMSE).

Methods to detect outliers

Box Plots and IQR method

Box plots

- Box plots are a visual method to identify outliers.
- Box plots is 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.
- IQR = Q3-Q1
- Lower Fence = Q1-1.5*IQR
- Upper Fence = Q3+1.5*IQR
- If any point lies below lower fence or above upper fence then we will consider them as outlier points.
- **Outlier detection based on Boxplots is univariate technique.**



Z-Score method

Z-score method

- Z-score method is another method for detecting outliers.
- This method is generally used when a variable's distribution looks close to Gaussian/Normal.
- Z-score is the number of standard deviations a value of a variable is away from the variable' 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 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.
- **Outlier detection based on Z-score method is univariate technique.**

Mahalanobis distance method

Mahalanobis distance

- The Mahalanobis distance is the distance between two points in a multivariate space.
- It’s often used to find outliers in statistical analyses that involve several variables.
- Mahalanobis Distance (MD) is an effective distance metric that finds the distance between the point and distribution.
- It works quite effectively on multivariate data because it uses a covariance matrix of variables to find the distance between data points and the center.
- This means that MD detects outliers based on the distribution pattern of data points, unlike the Euclidean distance.
- Any observation having p-value less than 0.001 will be considered as outlier.
- **Outlier detection based Mahalanobis distance method is multivariate technique.**

$$D^2 = (X_{p_1} - X_{p_2})^T . C^{-1} . (X_{p_1} - X_{p_2})$$

Isolation Forest

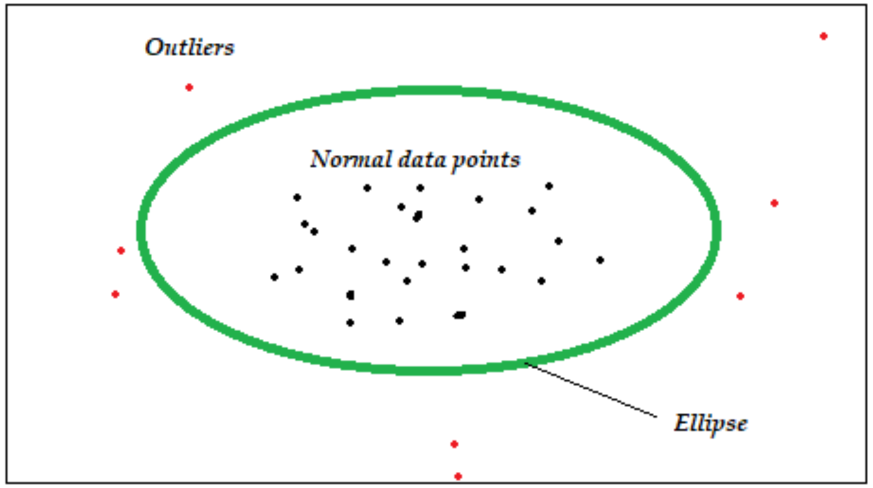
Isolation forest

- IsolationForests were built based on the fact that anomalies are the data points that are “few and different”.
- In an Isolation Forest, randomly sub-sampled data is processed in a tree structure based on randomly selected features.
- The samples that travel deeper into the tree are less likely to be anomalies as they required more cuts to isolate them.
- Similarly, the samples which end up in shorter branches indicate anomalies as it was easier for the tree to separate them from other observations.

Elliptic Envelope

Elliptic Envelope

- Elliptical Envelope— creates an imaginary elliptical area around a given dataset.
- Values that fall inside the envelope are considered normal data and anything outside the envelope is returned as outliers.



Column Standardization

Feature scaling

Feature Scaling

- Generally the dataset which we use for machine learning prediction have different scales.
- For example, one feature is entirely in kilograms while the other is in grams, another one is liters, and so on.
- In these scenarios it becomes difficult to compare them and our machine learning models also gets affected by these varying scale.
- One solution for this problem is feature scaling which brings those features on same scale and thus makes it feasible to compare them.

Example :

If you want to know whether it is warmer in Delhi or Shimla on a given day, and one is 68 degrees F and the other is 25 degrees C, you can’t just say 68 is bigger than 25 so Delhi is warmer. Instead, you need to reduce the measurements to the same scale, and then compare.

Why feature scaling?

Gradient Descent Based Algorithms

- Machine learning algorithms like **linear regression, logistic regression, neural network**, etc. that use gradient descent as an optimization technique require data to be scaled.

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

- **The presence of feature value X** in the formula will affect the step size of the gradient descent.
- To ensure that the gradient descent moves smoothly towards the minima and that the steps for gradient descent are updated at the same rate for all the features, we scale the data before feeding it to the model

Distance-Based Algorithms

- Distance algorithms like **KNN, K-means, and SVM** are most affected by the range of features. This is because behind the scenes **they are using distances between data points to determine their similarity**.
- This will impact the performance of the machine learning algorithm and obviously, we do not want our algorithm to be biased towards one feature.
- Therefore, we scale our data before employing a distance based algorithm so that all the features contribute equally to the result.

Normalization

Feature Scaling : Normalization

- 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.
- The formula for normalization is $$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$
- 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

Standardization

Feature Scaling : Standardization

- Standardization is another scaling technique where the values are centred 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.
- the formula for standardization: $X' = \frac{X - \mu}{\sigma}$

Where

μ = Mean of the feature values

σ = Standard deviation of the feature values

Normalize or Standardize?

- **Normalization** is **good to use** when you know that the distribution of your data **does not follow a Gaussian distribution**. This can be useful in algorithms that do not assume any distribution of the data like K-Nearest Neighbors and Neural Networks.
- **Standardization**, on the other hand, **can be helpful** in cases **where the data follows a Gaussian distribution**. However, this does not have to be necessarily true. Also, unlike normalization, standardization does not have a bounding range. So, even if you have outliers in your data, they will not be affected by standardization.
- However, at the end of the day, the choice of using normalization or standardization will depend on your problem and the machine learning algorithm you are using. **There is no hard and fast rule** to tell you when to normalize or standardize your data. You can always **start by fitting your model to raw, normalized and standardized data and compare the performance for best results.**

Treating Categorical Variable

Categorical Variable

- After handling missing values in the dataset, the next step in data analysis is to handle categorical data.
- Machine learning models require all input and output variables to be numeric.
- This means that if your data contains categorical data, you must encode it to numbers before you can fit and evaluate a model.
- Focusing only on numerical variables in the dataset isn’t enough to get good accuracy. Often categorical variables prove to be the most important factor and thus identify them for further analysis.

Types of categorical variable

A categorical or discrete variable is one that has two or more categories (values).

There are two different types of categorical variables

Nominal Variable

- A nominal variable has no intrinsic ordering to its categories.
- For example, gender is a categorical variable having two categories (Male and Female) with no inherent ordering between them.
- Another example is Country (India, Australia, America, and so forth).

Ordinal Variable

- An ordinal variable has a clear ordering within its categories.
- For example, consider temperature as a variable with three distinct (but related) categories (low, medium, high).
- Another example is an education degree (Ph.D., Master’s, or Bachelor’s).

Ordinal variable encoding

When the categorical variables are ordinal, the easiest approach is to replace each label/category by some ordinal number based on the ranks.

LabelEncoder

When the categories of an ordinal variable are naturally in ascending order then in that case we can use LabelEncoder to code that variable.

Example : If you have a variable denoting the shades of color of a product having three categories (dark, darker and darkest)

In this case we can use LabelEncoder to encode this ordinal variable.

Replace using the map function

This could be a very basic approach to manually replace categorical values with custom values.
Step 1: Create a dictionary with key as category and values with its rank.

Step 2: Create a new column and map the ordinal column with the created dictionary.

Step 3: Drop the original column.

Nominal variable encoding

One Hot Encoding

- For categorical variables where no ordinal relationship exists, the integer encoding may not be enough, at best, or misleading to the model at worst.
- In one Hot Encoding method, each category value is converted into a new column and assigned a value as 1 or 0 to the column.
- Forcing an ordinal relationship via an ordinal encoding **on nominal variable** and allowing the model to assume **a natural ordering between categories of nominal variable** may result in poor performance or unexpected results (predictions halfway between categories)

Feature Importance

Feature Importance

- Now a days with all of the packages and tools available, building a machine learning model isn’t difficult but ideally our objective is to build a good machine learning model which can help us make accurate prediction.
- **A huge thing that is often ignored is selecting the appropriate features for these models.** Useless data results in bias that messes up the final results of our machine learning.
- When we fit a supervised machine learning (ML) model, we often want to understand which features are most associated with our outcome of interest.
- Features that are highly associated with the outcome are considered more **important**.

Feature Importance

- Feature Importance refers to **techniques that calculate a score for all the input features for a given model** — the scores simply represent the “importance” of each feature.
- **A higher score means that the specific feature will have a larger effect on the model** that is being used to predict a certain variable.
- Suppose you have to buy a new house near your workplace. While purchasing a house, you might think of different factors.
- The most important factor in your decision making might be the location of the property, and so, you’ll likely only look for houses that are near your workplace.
- Feature importance works in a similar way, it will rank features based on the effect that they have on the model’s prediction.

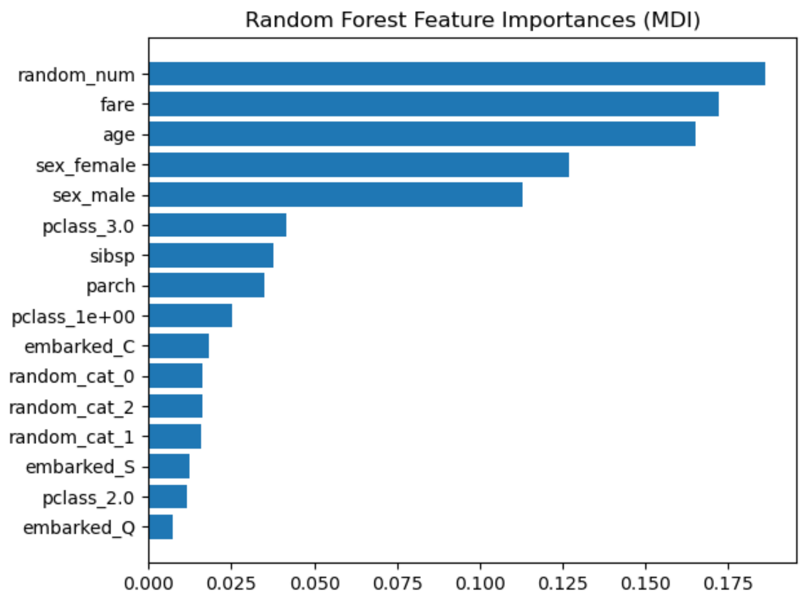
Calculate feature Importance

There are many different ways to calculate feature importance for different kinds of machine learning models.

Gini impurity

Gini importance is used to calculate the node impurity and feature importance is basically a reduction in the impurity of a node weighted by the number of samples that are reaching that node from the total number of samples.

The features are normalized against the sum of all feature values present in the tree and after dividing it with the total number of trees in our random forest, we get the overall feature importance.



Calculate feature Importance

Permutation Feature Importance

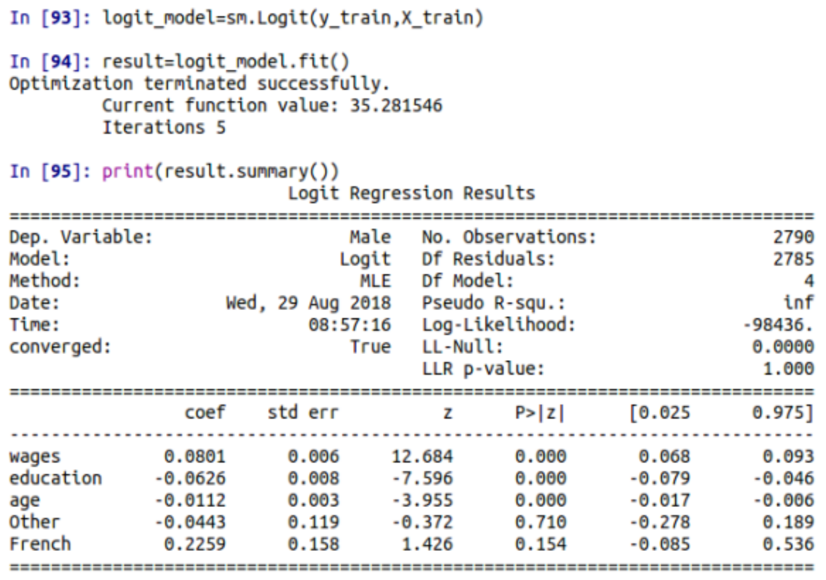
In this method The feature importance is calculated by noticing the increase or decrease in error when we permute the values of a feature.

If permuting the values causes a huge change in the error, it means the feature is important for our model.

The best thing about this method is that it can be applied to every supervised machine learning model.

Coefficient as feature importance

In case of linear model (Logistic Regression, Linear Regression, Regularization) we generally find coefficient to predict the output.



```
In [93]: logit_model=sm.Logit(y_train,X_train)

In [94]: result=logit_model.fit()
Optimization terminated successfully.
         Current function value: 35.281546
         Iterations 5

In [95]: print(result.summary())
                           Logit Regression Results
==============================================================================
Dep. Variable:                   Male   No. Observations:                 2790
Model:                          Logit   Df Residuals:                     2785
Method:                           MLE   Df Model:                            4
Date:                Wed, 29 Aug 2018   Pseudo R-squ.:                     inf
Time:                        08:57:16   Log-Likelihood:                -98436.
converged:                       True   LL-Null:                        0.0000
                                        LLR p-value:                     1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
wages          0.0801      0.006     12.684      0.000       0.068       0.093
education     -0.0626      0.008     -7.596      0.000      -0.079      -0.046
age           -0.0112      0.003     -3.955      0.000      -0.017      -0.006
Other         -0.0443      0.119     -0.372      0.710      -0.278       0.189
French         0.2259      0.158      1.426      0.154      -0.085       0.536
==============================================================================
```

Thank you!