Outlier Detection & Removal | How to Detect & Remove Outliers (Updated 2023)

CHIRAG GOYAL — Published On May 19, 2021 and Last Modified On April 14th, 2023

Beginner Data Cleaning Statistics Structured Data

Introduction

In my previous article, I talked about the theoretical concepts of outliers and tried to find the answer to the question: **"When should we drop outliers and when should we keep them?"**. In this article, I will focus on outlier detection and the different ways of treating them. It is important for a data scientist to find outliers and remove them from the dataset as part of the feature engineering before training machine learning algorithms for predictive modeling. Outliers present in a classification or regression dataset can lead to lower predictive modeling performance.

I recommend you read this **article** before proceeding so that you have a clear idea about the outlier analysis in Data Science Projects.

Learning Objectives

1. An Overview of outliers and why it's important for a data scientist to identify and remove them from data.
2. Undersand different techniques for outlier treatment: trimming, capping, treating as a missing value, and discretization.
3. Understanding different plots and libraries for visualizing and trating ouliers in a dataset.

This article was published as a part of the Data Science Blogathon

Table of Contents

How to Treat Outliers?

There are several ways to treat outliers in a dataset, depending on the nature of the outliers and the problem being solved. Here are some of the most common ways of treating outlier values.

Trimming

It excludes the outlier values from our analysis. By applying this technique, our data becomes thin when more outliers are present in the dataset. Its main advantage is its **fastest** nature.

Capping

In this technique, wecap our outliers data and make the limit i.e, above a particular value or less than that value, all the values will be considered as outliers, and the number of outliers in the dataset gives that capping number.

For example, if you're working on the income feature, you might find that people above a certain income level behave similarly to those with a lower income. In this case, you can cap the income value at a level that keeps that intact and accordingly treat the outliers.

Treating outliers as a missing value: Byassuming outliers as the missing observations, treat them accordingly, i.e., same as missing values imputation.

You can refer to the missing value article **here** .

Discretization

In this technique, by making the groups, we include the outliers in a particular group and force them to behave in the same manner as those of other points in that group. This technique is also known as **Binning**.

You can learn more about discretization **here** .

How to Detect Outliers?

For Normal Distributions

- Use empirical relations of Normal distribution.
- The data points that fall below ***mean-3*(sigma)*** or above ***mean+3*(sigma)*** are outliers, where mean and sigma are the **average value** and **standard deviation** of a particular column.


Characteristics of a Normal Distribution

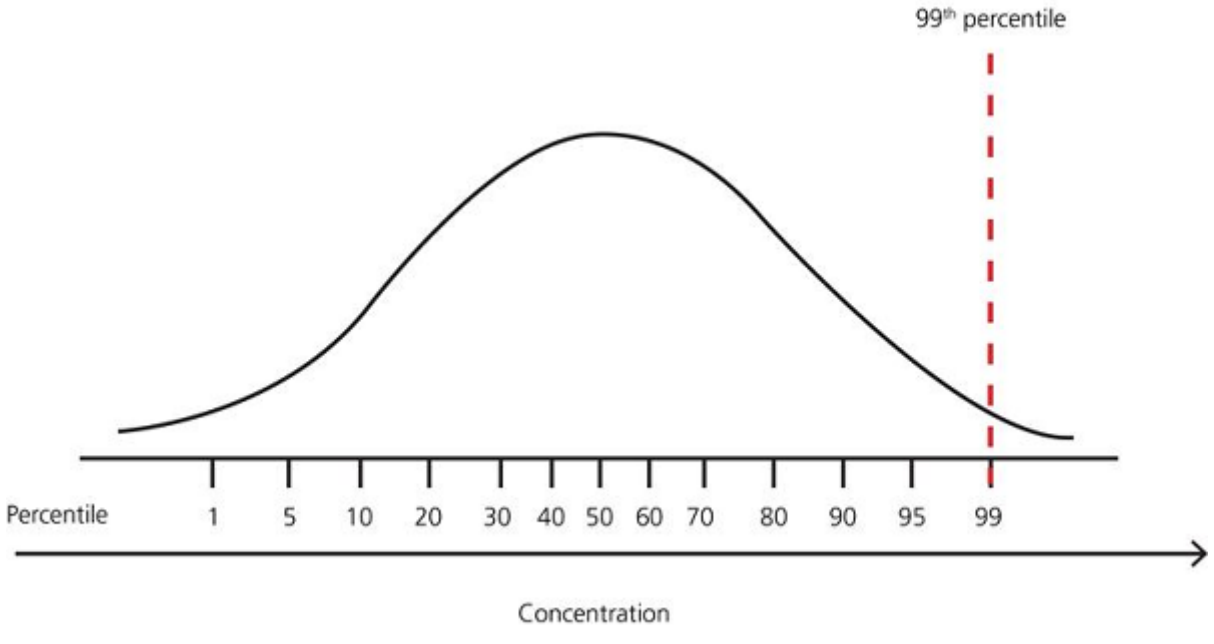
Source: sphweb.bumc.bu.edu

For Skewed Distributions

- Use Inter-Quartile Range (IQR) proximity rule.
- The data points that fall below ***Q1 – 1.5 IQR*** or above the third quartile ***Q3 + 1.5 IQR*** are outliers, where Q1 and Q3 are the **25th** and **75th percentile** of the dataset, respectively. IQR represents the inter-quartile range and is given by Q3 – Q1.

For Other Distributions

- Use **a** percentile-based approach.
- For Example, data points that are far from the 99% percentile and less than 1 percentile are considered an outlier.



Source: acutecaretesting.org

How to Detect and Remove Outliners in Python

Z-score Treatment

Assumption: The features are normally or approximately normally distributed.

1. **Step 1: Importing necessary dependencies**

   ```
   import numpy as np
   import pandas as pd
   import matplotlib.pyplot as plt
   import seaborn as sns
   ```

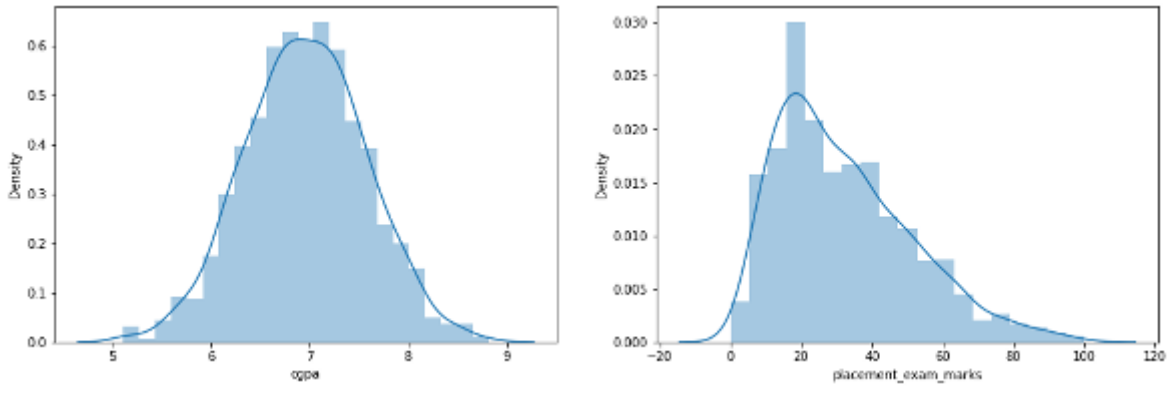
2. **Step 2: Read and load the dataset**

   ```
   df = pd.read_csv('placement.csv')
   df.sample(5)
   ```

	cgpa	placement_exam_marks	placed
6	6.69	11.0	0
19	7.63	29.0	0
258	6.12	6.0	1
606	6.79	24.0	0
675	6.10	20.0	0

3. **Step 3: Plot the distribution plots for the features**

   ```
   import warnings
   warnings.filterwarnings('ignore')
   plt.figure(figsize=(16,5))
   plt.subplot(1,2,1)
   sns.distplot(df['cgpa'])
   plt.subplot(1,2,2)
   sns.distplot(df['placement_exam_marks'])
   plt.show()
   ```



4. **Step 4: Finding the boundary values**

   print("Highest allowed",df['cgpa'].mean() + 3*df['cgpa'].std())

   print("Lowest allowed",df['cgpa'].mean() – 3*df['cgpa'].std())

   Output:

   Highest allowed 8.808933625397177

   Lowest allowed 5.113546374602842

5. **Step 5: Finding the outliers**

   df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]

6. **Step 6: Trimming of outliers**

   new_df = df[(df['cgpa'] < 8.80) & (df['cgpa'] > 5.11)]

   new_df

7. **Step 7: Capping on outliers**

   upper_limit = df['cgpa'].mean() + 3*df['cgpa'].std()

   lower_limit = df['cgpa'].mean() – 3*df['cgpa'].std()

8. **Step 8: Now, apply the capping**

   df['cgpa'] = np.where(

   df['cgpa']>upper_limit,

   upper_limit,

   np.where(

   df['cgpa']<lower_limit,

   lower_limit,

   df['cgpa']

9. **Step 9: Now, see the statistics using the "Describe" function**

   df['cgpa'].describe()

Output:

```
count    1000.000000
mean        6.961499
std         0.612688
min         5.113546
25%         6.550000
50%         6.960000
75%         7.370000
max         8.808934
Name: cgpa, dtype: float64
```

This completes our Z-score-based technique!

IQR Based Filtering

Used when our data distribution is skewed.

Step-1: Import necessary dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Step-2: Read and load the dataset

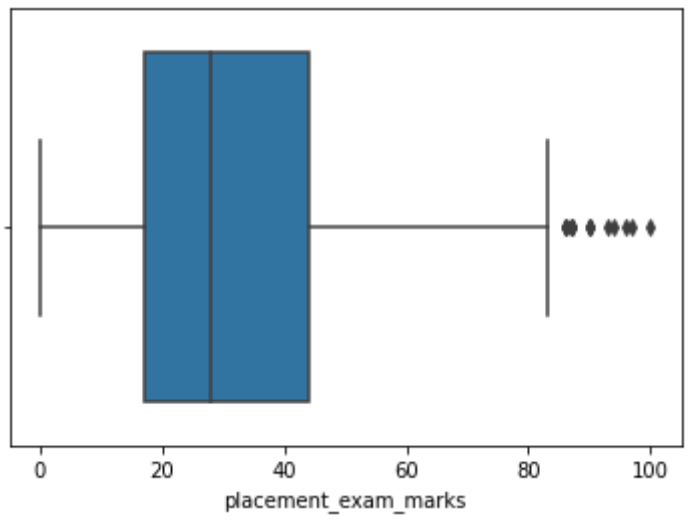
```
df = pd.read_csv('placement.csv')
df.head()
```

Step-3: Plot the distribution plot for the features

```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(df['cgpa'])
plt.subplot(1,2,2)
sns.distplot(df['placement_exam_marks'])
plt.show()
```

Step-4: Form a box-plot for the skewed feature

```
sns.boxplot(df['placement_exam_marks'])
```



Step-5: Finding the IQR

```
percentile25 = df['placement_exam_marks'].quantile(0.25)
percentile75 = df['placement_exam_marks'].quantile(0.75)
```

Step-6: Finding the upper and lower limits

```
upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
```

Step-7: Finding outliers

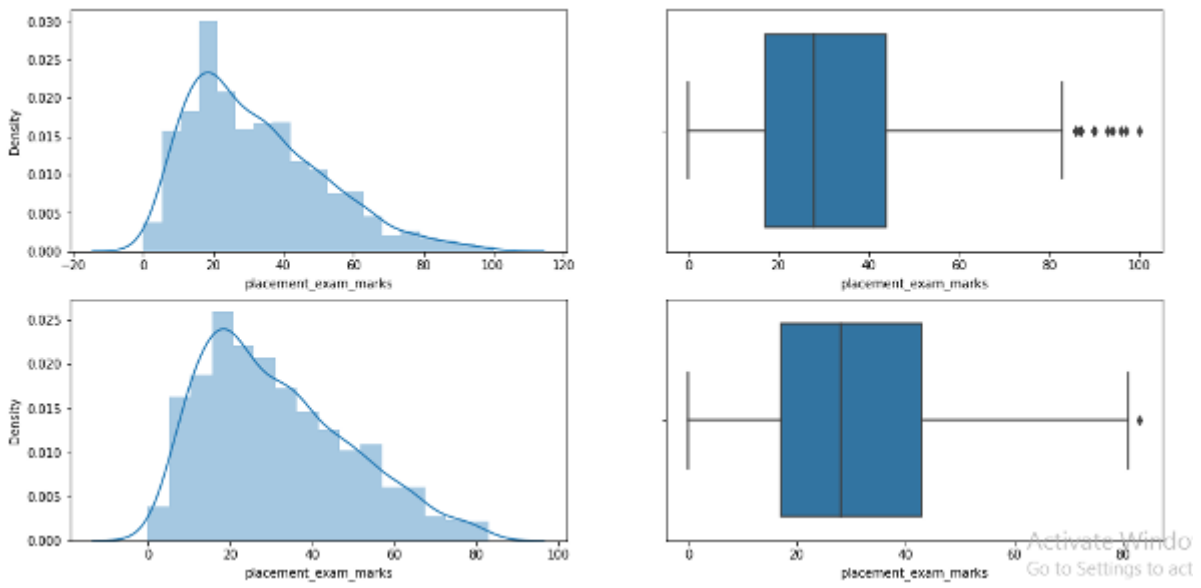
```
df[df['placement_exam_marks'] > upper_limit]
df[df['placement_exam_marks'] < lower_limit]
```

Step-8: Trimming outliers

```
new_df = df[df['placement_exam_marks'] < upper_limit]
new_df.shape
```

Step-9: Compare the plots after trimming

```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['placement_exam_marks'])
plt.subplot(2,2,2)
sns.boxplot(df['placement_exam_marks'])
plt.subplot(2,2,3)
sns.distplot(new_df['placement_exam_marks'])
plt.subplot(2,2,4)
sns.boxplot(new_df['placement_exam_marks'])
plt.show()
```

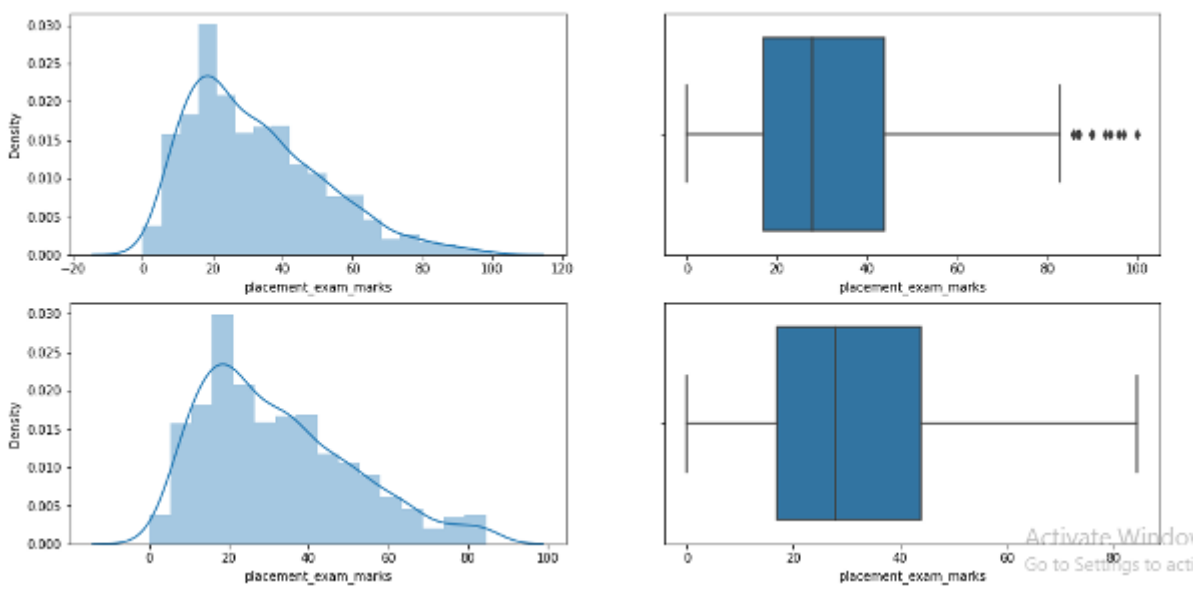


Step-10: Capping

```
new_df_cap = df.copy()
new_df_cap['placement_exam_marks'] = np.where(
    new_df_cap['placement_exam_marks'] > upper_limit,
    upper_limit,
    np.where(
        new_df_cap['placement_exam_marks'] < lower_limit,
        lower_limit,
        new_df_cap['placement_exam_marks']
```

Step-11: Compare the plots after capping

```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(df['placement_exam_marks'])
plt.subplot(2,2,2)
sns.boxplot(df['placement_exam_marks'])
plt.subplot(2,2,3)
sns.distplot(new_df_cap['placement_exam_marks'])
plt.subplot(2,2,4)
sns.boxplot(new_df_cap['placement_exam_marks'])
plt.show()
```



This completes our IQR-based technique!

Percentile Method

- This technique works by setting a particular threshold value, which is decided based on our problem statement.
- While we remove the outliers using capping, then that particular method is known as **Winsorization**.
- Here, we always maintain **symmetry** on both sides, meaning if we remove 1% from the right, the left will also drop by 1%.

Steps to follow for the percentile method:

Step-1: Import necessary dependencies

```
import numpy as np
import pandas as pd
```

Step-2: Read and Load the dataset

```
df = pd.read_csv('weight-height.csv')
df.sample(5)
```

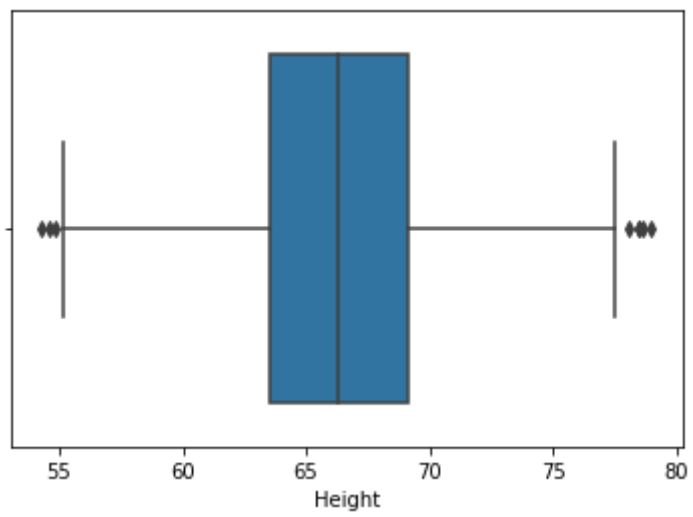
	Gender	Height	Weight
0	Male	73.847017	241.893563
1	Male	68.781904	162.310473
2	Male	74.110105	212.740856
3	Male	71.730978	220.042470
4	Male	69.881796	206.349801

Step-3: Plot the distribution plot of the “height” feature

```
sns.distplot(df['Height'])
```

Step-4: Plot the box-plot of the “height” feature

```
sns.boxplot(df['Height'])
```



Step-5: Finding the upper and lower limits

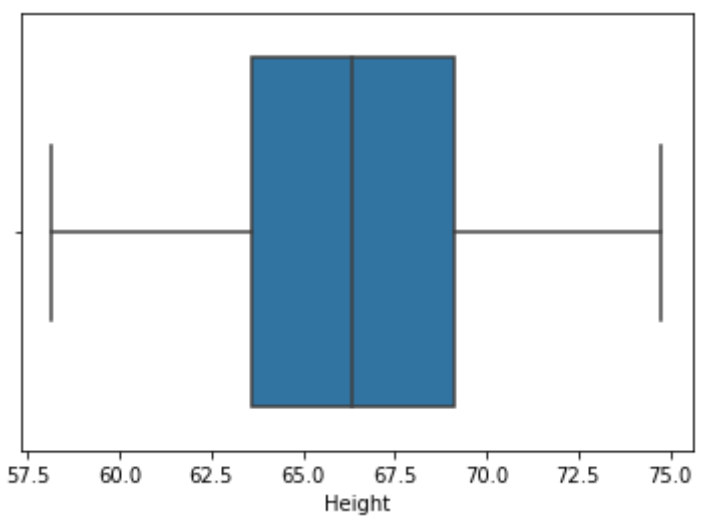
```
upper_limit = df['Height'].quantile(0.99)
lower_limit = df['Height'].quantile(0.01)
```

Step-6: Apply trimming

```
new_df = df[(df['Height'] <= 74.78) & (df['Height'] >= 58.13)]
```

Step-7: Compare the distribution and box-plot after trimming

```
sns.distplot(new_df['Height'])
sns.boxplot(new_df['Height'])
```



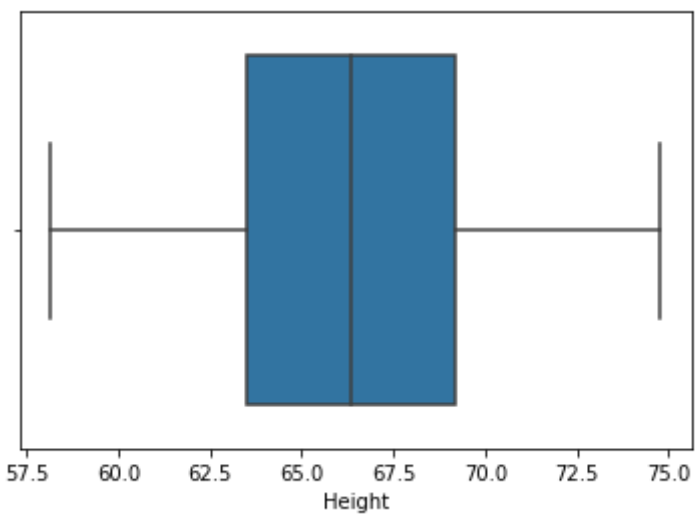
Winsorization

Step-8: Apply Capping (Winsorization)

```
df['Height'] = np.where(df['Height'] >= upper_limit,
        upper_limit,
        np.where(df['Height'] <= lower_limit,
        lower_limit,
        df['Height']))
```

Step-9: Compare the distribution and box-plot after capping

```
sns.distplot(df['Height'])
sns.boxplot(df['Height'])
```



This completes our percentile-based technique!

Conclusion

Outlier detection and removal is a crucial data analysis step for a machine learning model, as outliers can significantly impact the accuracy of a model if they are not handled properly. The techniques discussed in this article, such as Z-score and Interquartile Range (IQR), are some of the most popular methods used in outlier detection. The technique to be used depends on the specific characteristics of the data, such as the distribution and number of variables, as well as the required outcome.

Key Takeaways

- Outliers can be treated in different ways, such as trimming, capping, discretization, or by treating them as missing values.
- Emperical relations are used to detect outliers in normal distributions, and Inter-Quartile Range (IQR) is used to do so in skewed distributions. For all other distributions, we use the percentile-based approach.
- Z-score treatment is implemented in Python by importing the necessary dependencies, reading and loading the dataset, plotting the distribution plots, finding the boundary values, finding the outliers, trimming, and then capping them.

Frequently Asked Questions

Q1. What are some of the most popular outlier detection techniques?

A. Most popular outlier detection methods are Z-Score, IQR (Interquartile Range), Mahalanobis Distance, DBSCAN (Density-Based Spatial Clustering of Applications with Noise, Local Outlier Factor (LOF), and One-Class SVM (Support Vector Machine).

Q2. What are the libraries and plots we can utilize to detect and remove outliers in a data set for a data science project?

A. Libraries like SciPy and NumPy can be used to identify outliers. Also, plots like Box plot, Scatter plot, and Histogram are useful in visualizing the data and its distribution to identify outliers based on the values that fall outside the normal range.

Q3. What is the advantage of removing outliers?

A. The benefit of removing outliers is to enhance the accuracy and stability of statistical models and ML algorithms by reducing their impact on results. Outliers can distort statistical analyses and skew results as they are extreme values that differ from the rest of the data. Removing outliers makes the results more robust and accurate by eliminating their influence. It reduces overfitting in ML algorithms by avoiding fitting to extreme values instead of the underlying data pattern.

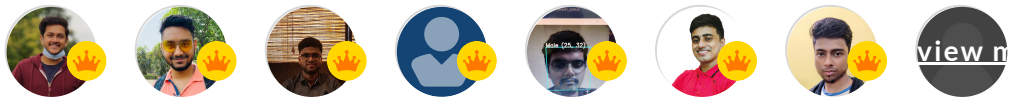
blogathon Outlier Detection python statistics

About the Author

CHIRAG GOYAL

I am currently pursuing my Bachelor of Technology (B.Tech) in Computer Science and Engineering from the Indian Institute of Technology Jodhpur(IITJ). I am very enthusiastic about Machine learning, Deep Learning, and Artificial Intelligence. Feel free to connect with me on Linkedin.

Our Top Authors



Previous Post

Stock Price Prediction and Forecasting using Stacked LSTM.

Next Post

Dictionaries 101 – A Super Guide for a dictionaries in Python for Absolute Beginners

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

Notify me of follow-up comments by email.

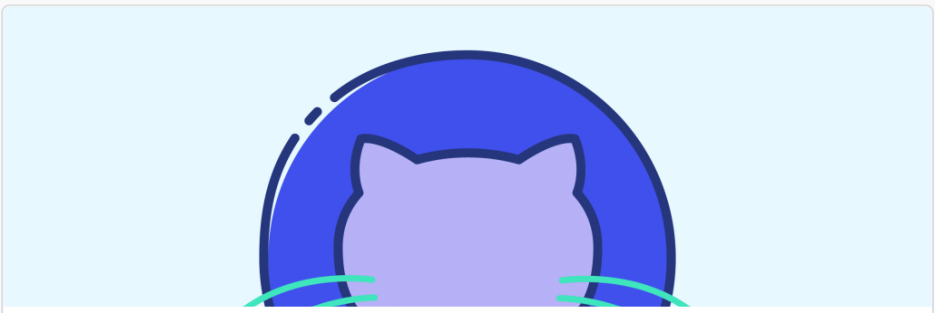
Notify me of new posts by email.

Submit

Top Resources

DragGAN: Google Researchers Unveil AI Technique for Magical Image Editing

Yana Khare - MAY 22, 2023



Top 10 GitHub Data Science Projects For Beginners

avcontentteam - MAY 22, 2023

Understand Random Forest Algorithms With Examples (Updated 2023)

Sruthi E R - JUN 17, 2021

Chatgpt-4 v/s Google Bard: A Head-to-Head Comparison

Gyan Prakash Tripathi - MAY 11, 2023

Analytics Vidhya

Download App

Analytics Vidhya

About Us

Our Team

Careers

Contact us

Companies

Post Jobs

Trainings

Hiring Hackathons

Advertising

Data Scientists

Blog

Hackathon

Discussions

Apply Jobs

Visit us