­CCT College Dublin

|  |  |
| --- | --- |
| Module Titles: | Data Preparation & Visualisation, Machine Learning for Data Analytics, Programming for Data Analytics, Statistics for Data Analytics |
| Assessment Title: | Integrated CA, Ireland Tourism |
| Lecturer Names: | David McQuaid, Dr. Muhammad Iqbal, Sam Weiss, Taufique Ahmed |
| Student Full Name: | Ricardo Alves de Souza |
| Student Number: | 2018499 |
| Assessment Due Date: | 07/04/2024 |
| Date of Submission: | 12/04/2024 |

Declaration

|  |
| --- |
| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |

Title: Integrated CA, Ireland Tourism

Author: Ricardo Alves de Souza

E-mail: [2018499@student.cct.ie](mailto:2018499@student.cct.ie?subject=Integrated%20CA1)

GitHub Address: <https://github.com/ricardoasouz/2024---MSc-in-Data-Analytics---Feb---FT/tree/main/MSC_DA_CA1>

Table of Contents

[Abstract 4](#_Toc163754781)

[Introduction 4](#_Toc163754782)

[EDA Exploratory Data Analysis 5](#_Toc163754783)

[Obtaining The Data 5](#_Toc163754784)

[Observing the “.CSV” file 5](#_Toc163754785)

[Preparing the Dataset for Machine Learning and Statistics 7](#_Toc163754786)

[Plotting the Dataset 12](#_Toc163754787)

[STATISTICS 15](#_Toc163754788)

[Descriptive Statistics 15](#_Toc163754789)

[Count or size 15](#_Toc163754790)

[Minimum 15](#_Toc163754791)

[Maximum 16](#_Toc163754792)

[Sum 16](#_Toc163754793)

[Mean 17](#_Toc163754794)

[Median 17](#_Toc163754795)

[Range 18](#_Toc163754796)

[Midrange (MR) 19](#_Toc163754797)

[Variance 19](#_Toc163754798)

[Standard deviation 20](#_Toc163754799)

[Quartiles 20](#_Toc163754800)

[Interquartile Range (IQR) 21](#_Toc163754801)

[Outliers 22](#_Toc163754802)

[Mean absolute deviation (MAD) 23](#_Toc163754803)

[Mode 24](#_Toc163754804)

[Skewness 25](#_Toc163754805)

[Kurtosis ( Pearson Kurtosis ) 27](#_Toc163754806)

[Kurtosis Excess ( Fischer Kurtosis ) 27](#_Toc163754807)

[Normal distribution 28](#_Toc163754808)

[Reference list 34](#_Toc163754809)

­

[Figure 1 - Table of Some Descriptive Statistics 7](#_Toc162356614)

­­

# Abstract

*An initial examination of the dataset in data science is crucial for gaining insights into its features, uncovering interconnections, detecting emergent trends, and conditioning the dataset for more advanced analytical processes, which might include predictive modelling or time series forecasting.*

# Introduction

An initial examination of the dataset in data science is crucial for gaining insights into its features, uncovering interconnections, detecting emergent trends, and conditioning the dataset for more advanced analytical processes, which might include predictive modelling or time series forecasting

# EDA Exploratory Data Analysis

Exploratory Data Analysis is an important process performed in data science to understand the data and its characteristics, to identify correlations, to extract patterns and to prepare the data for further stages of analysis such as forecasting or prediction. (Aloorravi, 2024)

## Obtaining The Data

This study has acquired data to analyse from the website of the government of Ireland, “CSO Central Statistics Office” in March of 2024.

https://www.cso.ie/en/statistics/tourismandtravel/

The segment chosen for analysis was:

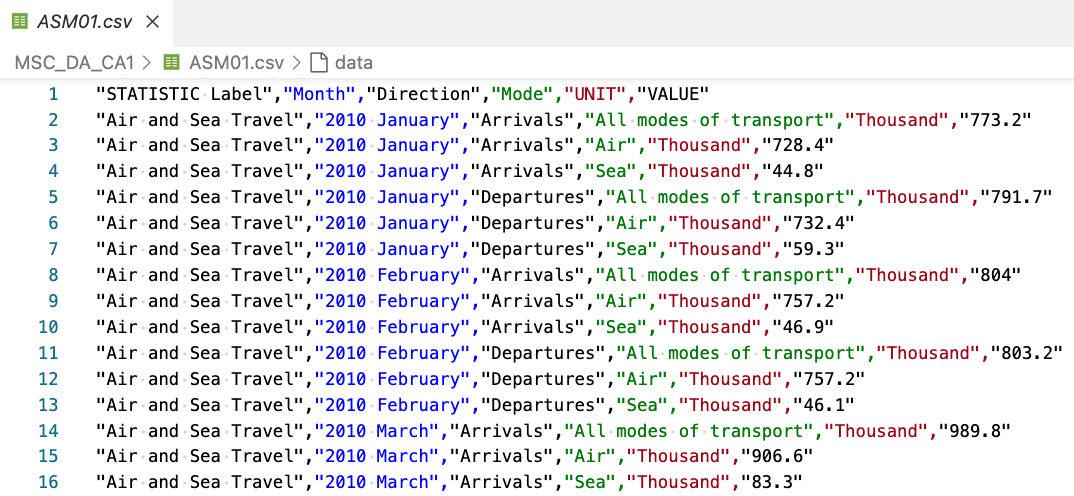
“Air and Sea Travel Statistics”

The file can be obtained from this internet address: “https://data.cso.ie/table/ASM01”, in multiple formats like, (“CSV 1.0”, “JSON-stat 2.0”, “JSON-stat 1.0”, “PX 2013”, “XLSX 2007”).

The format chosen in this case is “CSV 1.0”, an acronym for Comma Separated Values, and the file are “ASM01.csv”, Air and Sea Travel.

This dataset brings to us quantities of Arrives and Departures by Air and Sea in Ireland in the period about January 2010 to January 2024, separated by month with your values in thousands.

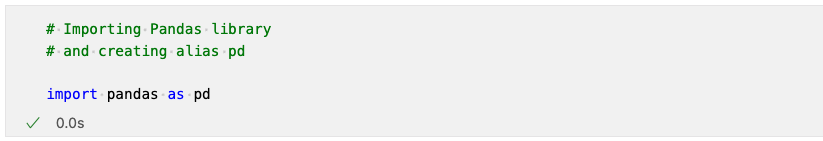
## Observing the “.CSV” file



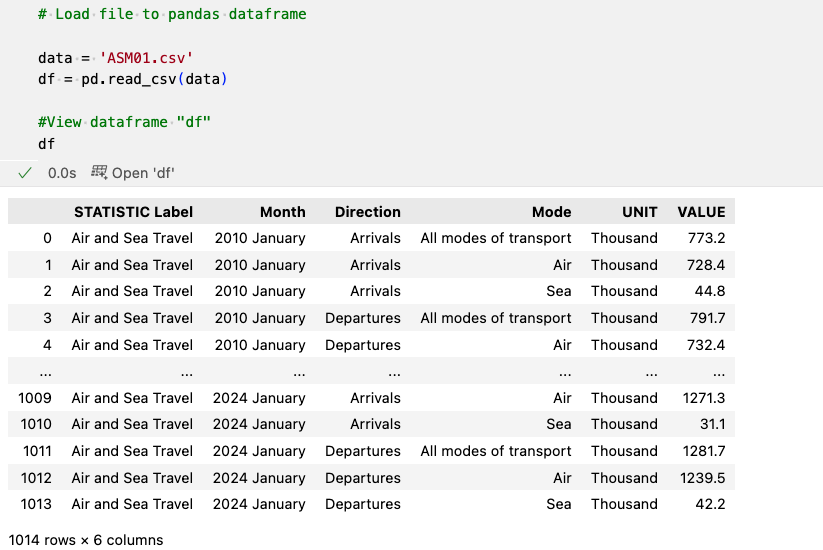
The ASM01.csv came with 1015 lines, with this structure can be seen in the previous picture. The first line with labels of columns: “STATISTIC Label”, “Month”, “Direction”, “Mode”, “UNIT”, “VALUE”, and on sequence your respective rows with values.

Python is the computer program language chosen for this study of the case, with code in the format of Jupiter Notebook and with a complement of some libraries like Pandas, NumPy, MatPlotLib, SciPy, SeaBorn…

In a “Jupiter Notebook”, let’s start importing “Pandas” with a keyword from Python “as” to create an alias name “pd”.

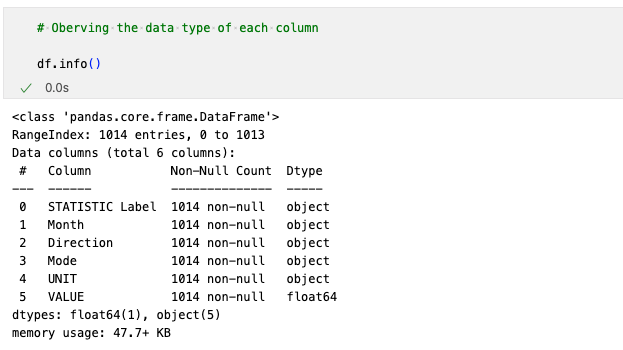
­­

With Pandas imported let’s load the “ASM01.CSV” file and for this make use of method “read\_csv()”, creating a new data frame “df” and viewing the result.



­ With the file already loaded to a “Pandas” data frame can view the labels, the five first rows and the last five rows of the data frame and message “1014 rows x 6 columns”. He just got one line from “ASM01.csv” to create the labels of columns.

Now let’s observe more details from the data frame, like the type of data in each column and if have a presence of “null values”, utilizing the “info()” method from “Pandas” for this.



It is possible to view that 5 columns are in the categorical format “object”, just one column “VALUE”, is in numerical format “float64”. And all columns don’t have any null value, “non-null”.

To perform statistical calculations, summarization and or start plot graphics of this dataset it needs to organize, remodel and convert these values to a numerical format in a way to be more easier to do calculations.

## Preparing the Dataset for Machine Learning and Statistics

In his book “The Visual Display of Quantitative Information”, Edward Tufte discusses his famous six principles of graphical integrity. The six principles of graphical integrity are (comparisons, causality, multivariate, integration, documentation, and context).

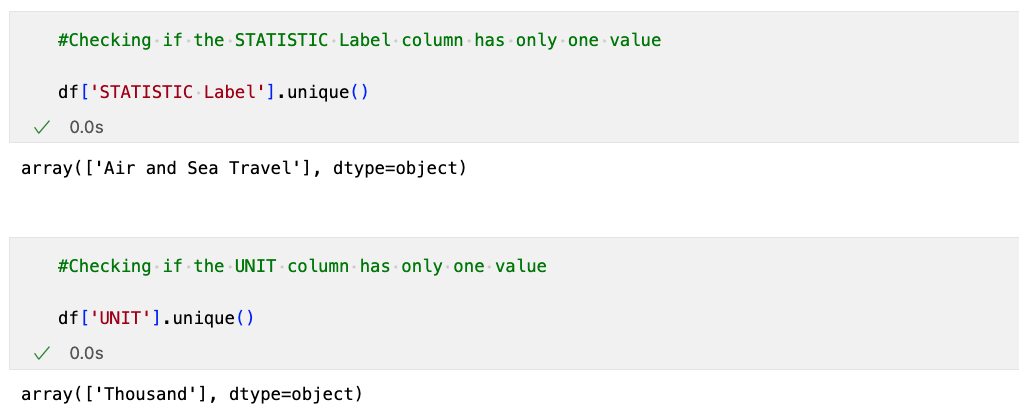
Also, he describes the best practices for data visualization:

* Above all else, show the data. Create the simplest graph that conveys the information you want to present.
* Maximize the data-ink ratio. Every bit of ink requires a reason. ...
* Erase non-data-ink.
* Erase redundant data-ink.
* Revise and edit. (Tufte, 1997)

Let’s try to prepare this data set keeping these principles in mind.

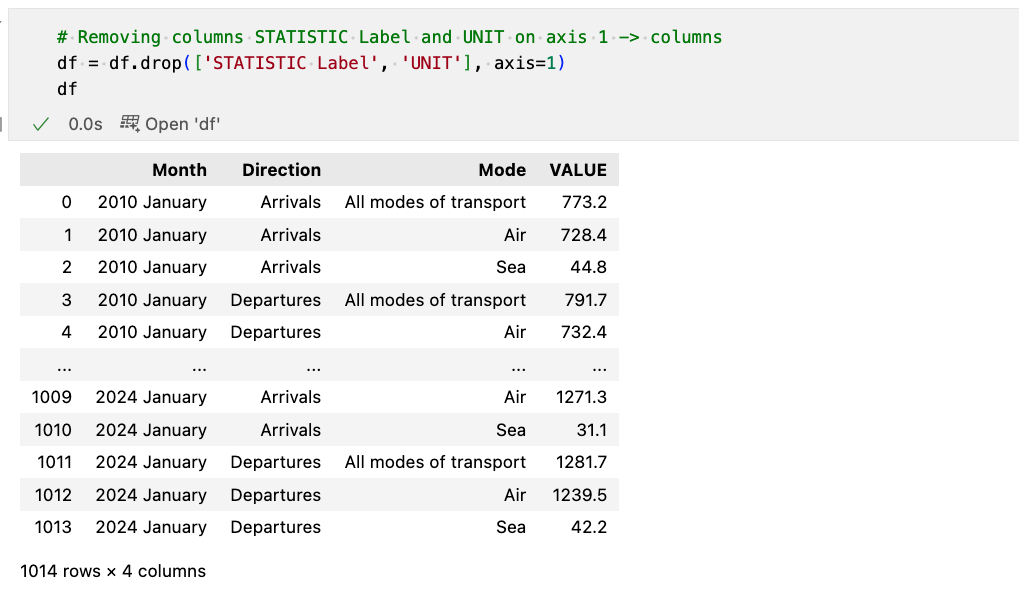
First of all, let’s confirm if the columns “STATISTIC Label” and “UNIT” have just the same value for all rows, making them “Air and Sea Travel” and “Thousand” redundant values.

For this, it’s possible to utilize the method “unique()”, he answers us with an array of unique values found in the column.



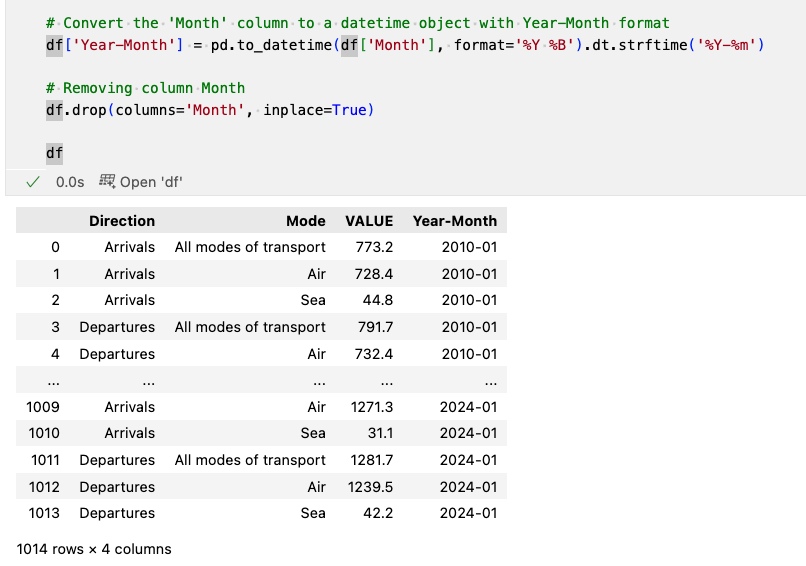
The answer is just one value on the array making these values redundant. But is interesting to not lose this important information “Thousand”, meaning the scale of values, let’s move this to the title of the labels of columns later.

Let’s perform the remotion of these columns “STATISTIC LABEL” and “UNIT”, utilizing the method “drop()”, starting to clean the dataset and visualizing the results.

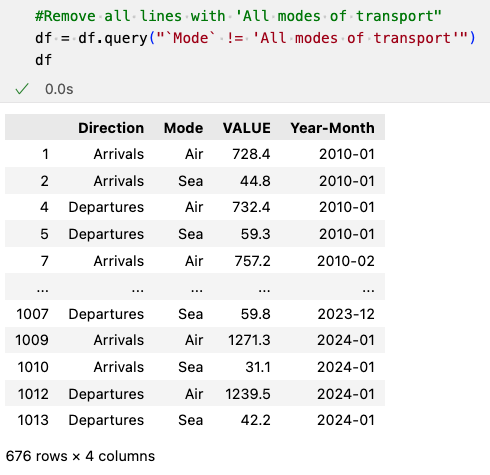


The result is now the data frame has only four columns, but still with too many categorical values with the description of column “VALUE” beside the same row.

The column “Month” can be transformed into the “DateTime” type, for this let’s utilize the method “to\_datetime()” of Pandas converting the values to the string format “Year-Month”, perfect to visualize without the “day” redundant information on this case, and removing the original column “Month” after this.



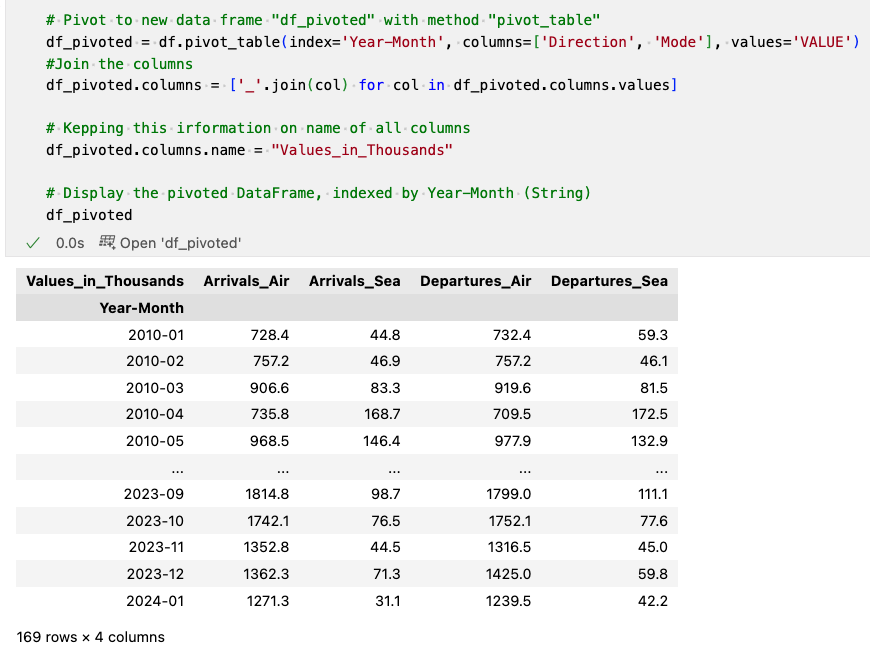
The column “Mode” has rows with the value “All modes of transport” and the “VALUE” column has an apparent sum of the two rows below but with some round differences. This sum is redundant and unnecessary at the moment and easy to have later by calc. Let’s remove all rows with the value “All modes of transport” from the data frame. For this let’s use the method “query()” from “Pandas” It works similarly to a query in “SQL Language”, in this case filtering and returning only rows when column “mode” is different “!=” to value “All modes of transport”.



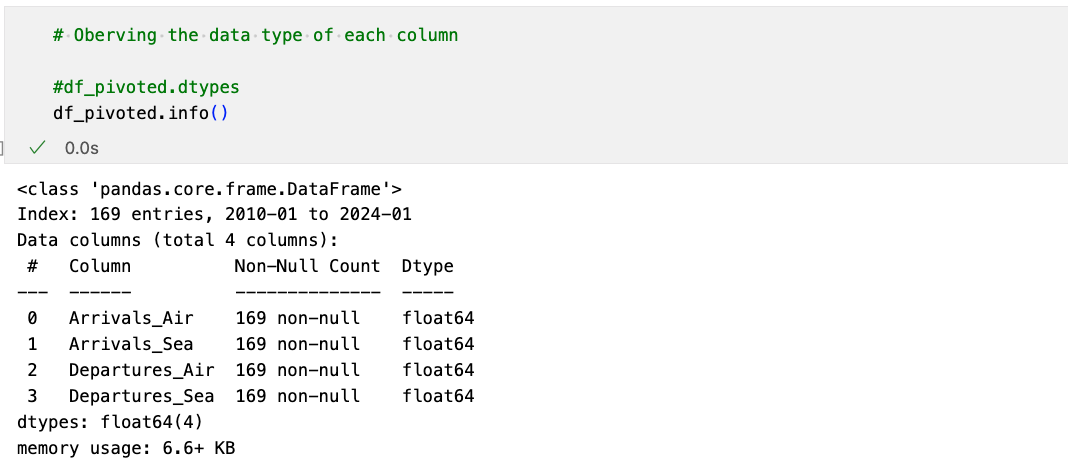
After deleting the rows with “All modes of transport”, the number of rows reduced from 1014 to 676.

The column “Year-Month” can start to be the index of the data set, making every month one row and all these values on the “Direction” and “Mode” columns can be joined e.g., “Arrivals” + “Air”, starting to be news columns in one new data frame, and your respective values from the column “VALUE”, below of the column created.

For this let’s utilize the method “pivot\_table()” passing these specific parameters and create then a new data frame called “df\_pivoted”.

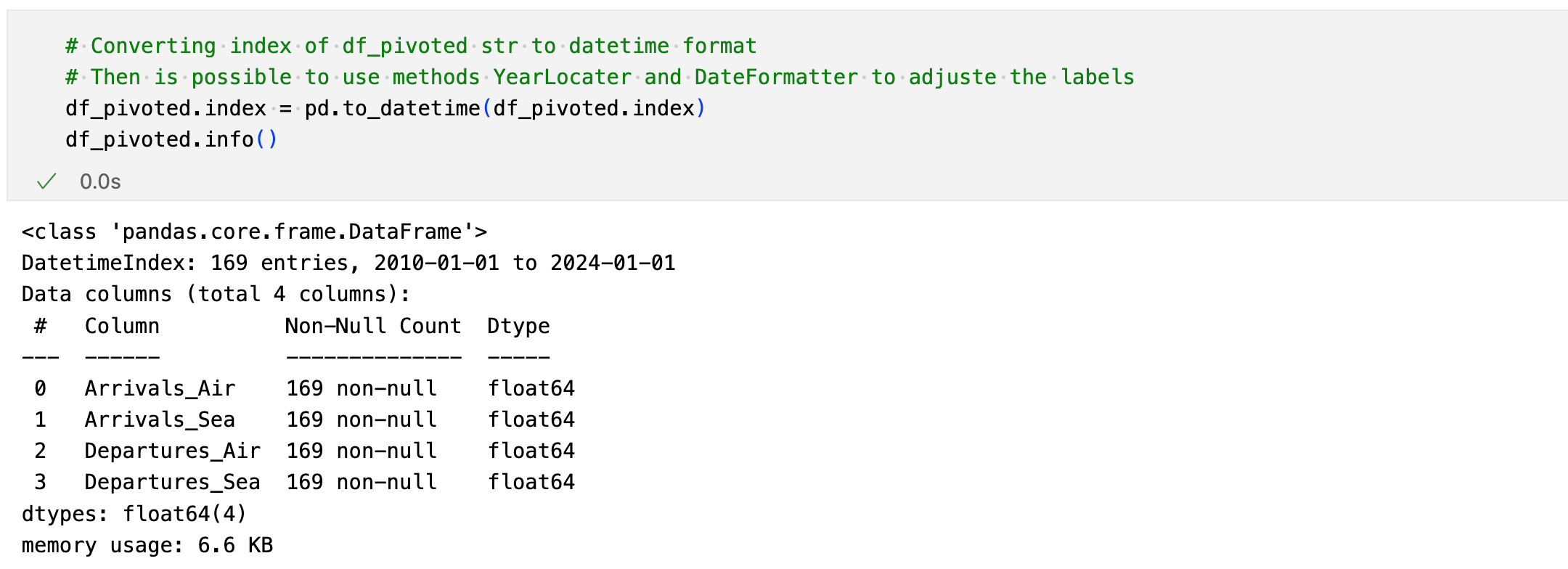


Let’s use the method “info()” on the new data frame “df\_pivoted” to check some of your parameters.

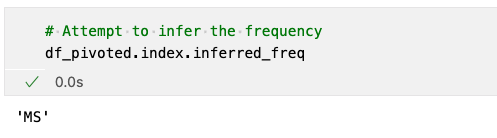


Now is possible to view all columns is on the numerical format “float64”, the categorical values before started to be labels of the columns and the index is still in string format. Also, the number of entries/rows was reduced to 169, representing each month between 2010-01 and 2024-01.

Let’s change the index to “DateTime” format then it’s possible to utilize methods like “YearLocator” and “DateFormatter” to simplify adjustments on labels.



With the index in the format “DateTime” is possible to check and certify your frequency. For this let’s utilize the method “inferred\_freq”, from “Pandas”.



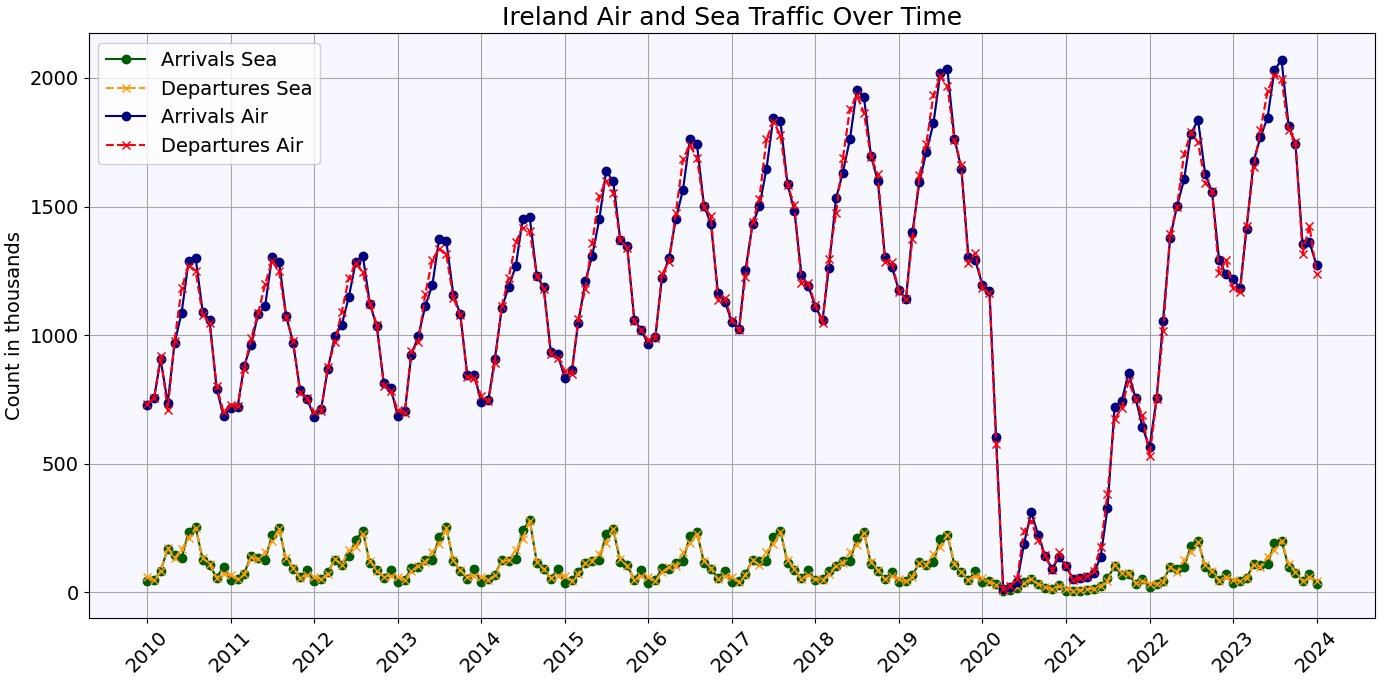
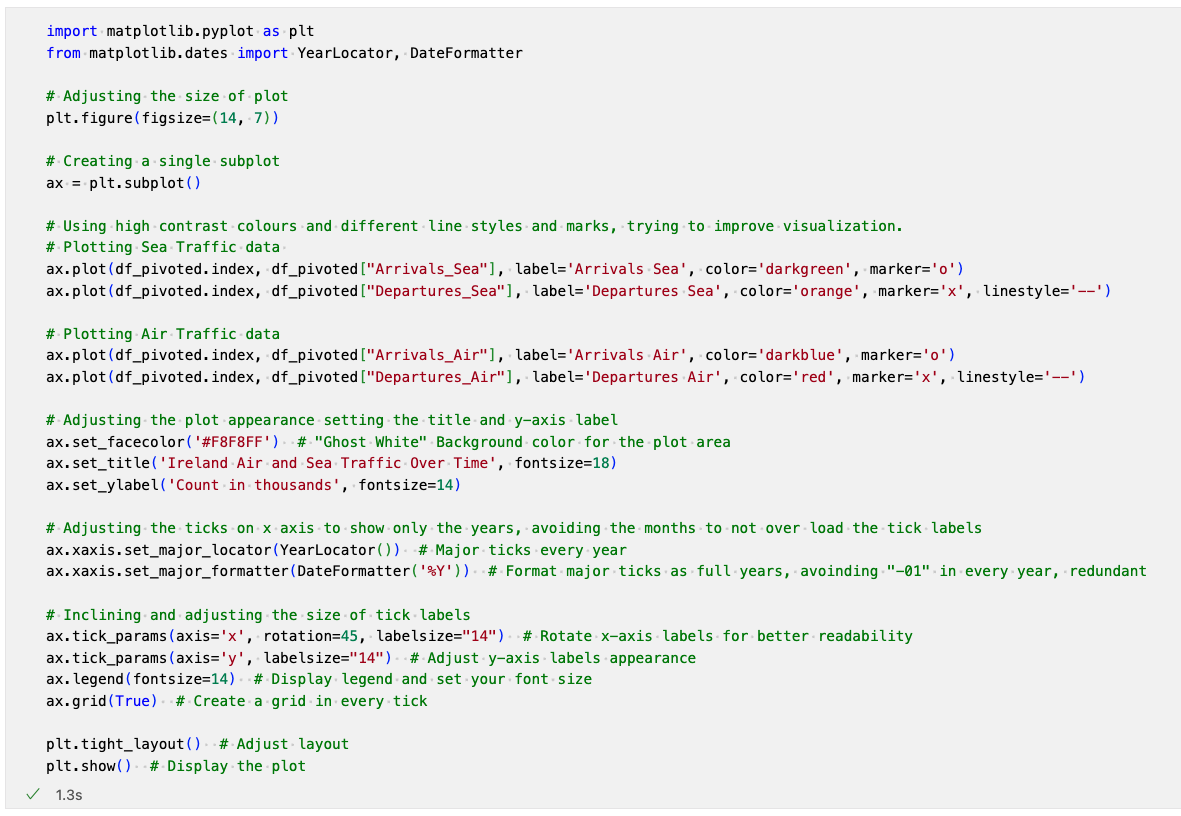
The return “MS” means “Month Sequence”, this confirms to us that all rows on the data frame are in a perfect chronological sequence of months.

Now we have a new data frame with the same content but indexed in date and time format and organized in numeric columns with your referent categorical values on the labels of these columns, making then easier to plot graphs, perform statistics calculations and apply machine learning models.

Let’s plot this new data set cleaned and prepared and check what can be viewed.

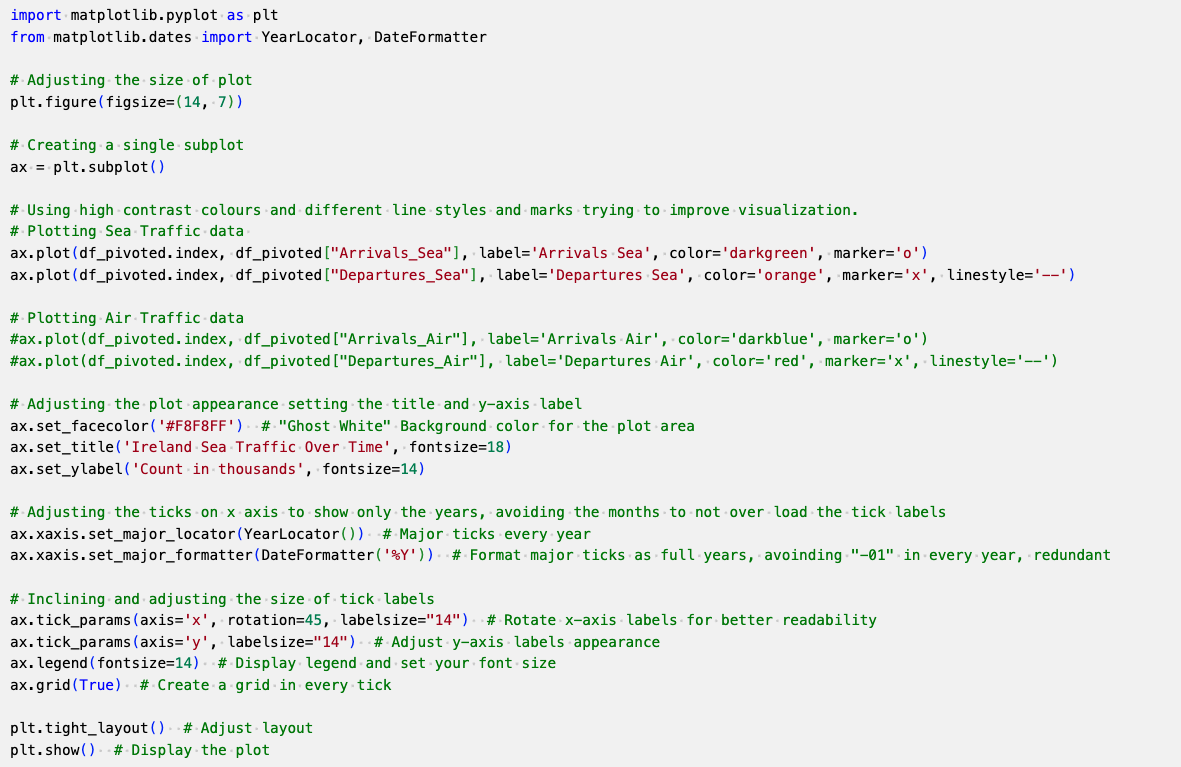
For this let’s import the library “matplotlib.pyplot” and set your alias name as “plt” and from “matplotlib.dates” let’s import “YearLocator” and “DateFormatter”, these libraries are going to help us adjust the labels on graph plotted. Comments in the code “#” have more details and explanations about each method and parameter.

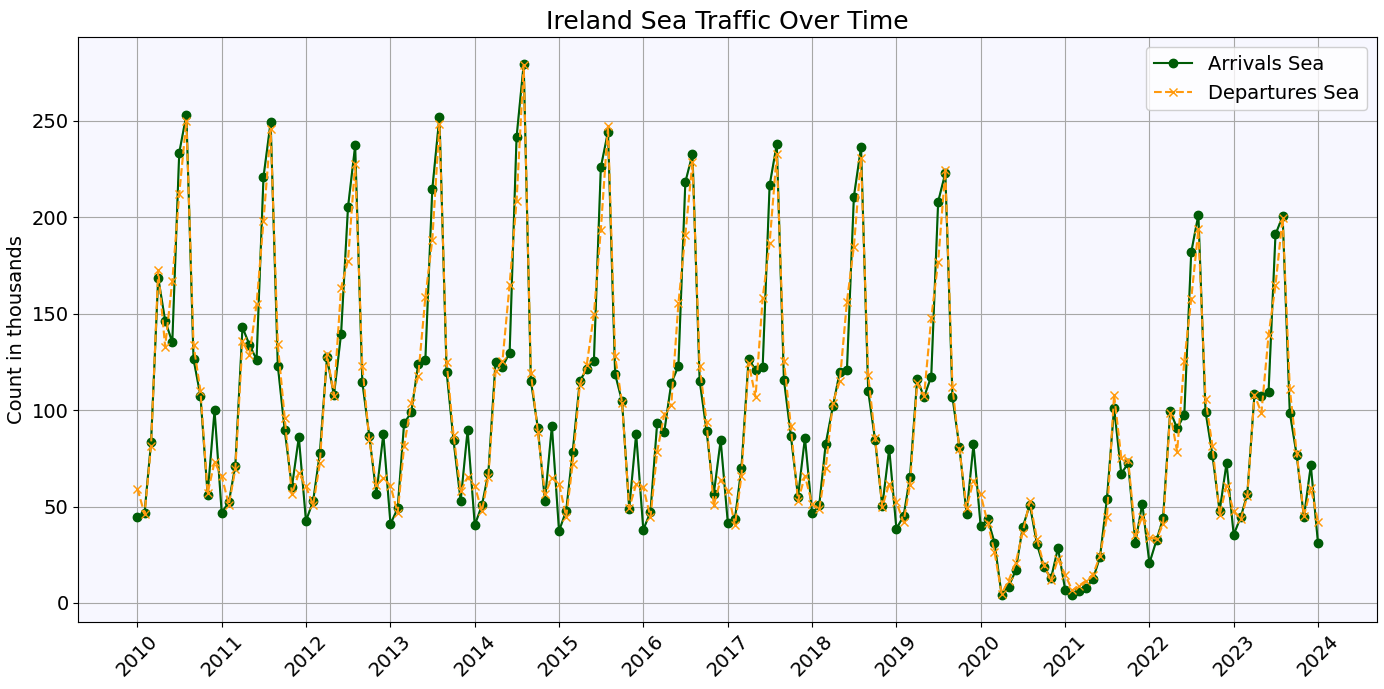
## Plotting the Dataset

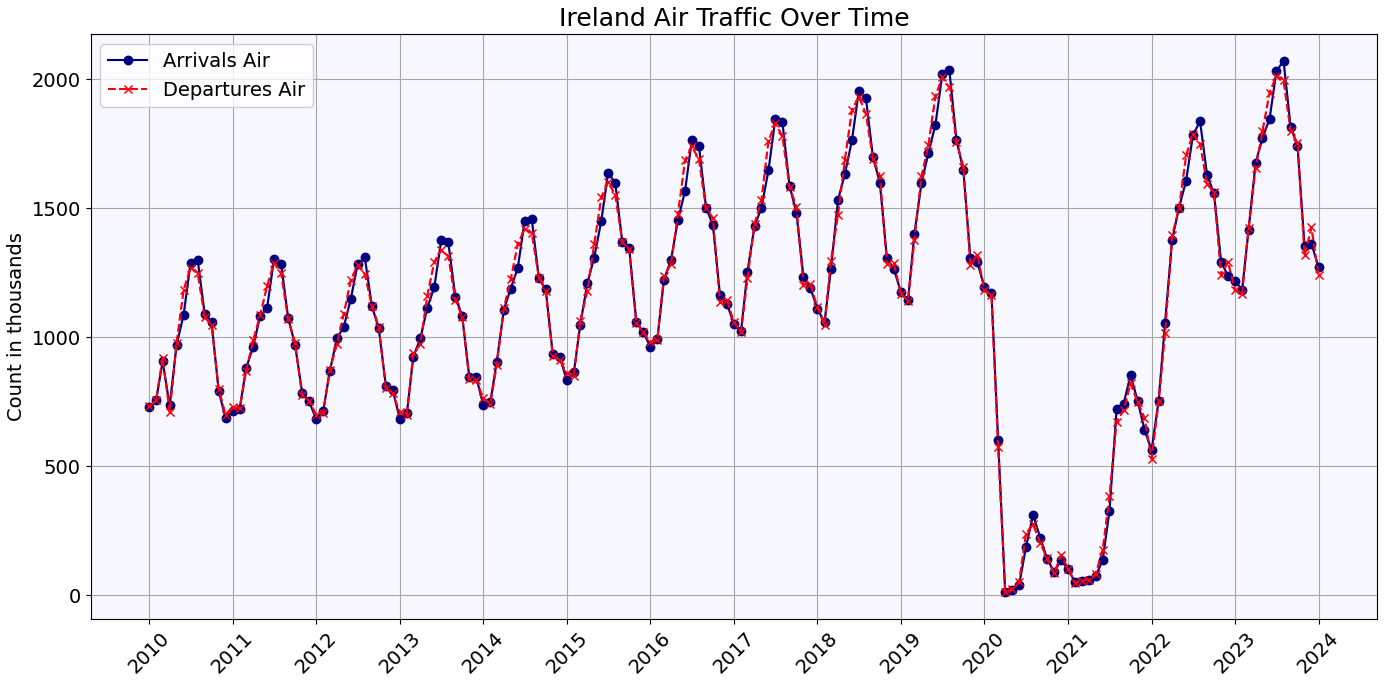


The full data is plotted on the graph above and the first thing possible to view is the anomaly caused by the COVID-19 pandemic, in the air sector around the year 2020 but keeping the plots on the same scale makes it harder to view what exactly happened in the sea sector.

Let’s plot the “Air traffic” separated from “Sea traffic” to view what can see. For this let’s just comment with “#” the respective lines on code from each other.







With the graphics plotted separately, it is easy to view the “Sea traffic” as the same anomaly after the year 2020, because the scale is now adjusted for each one.

It is also possible to see that “Sea traffic” is decreasing in this sample, and the variation fluctuates from a peak of 250 thousand in mid-2010 to 200 thousand in mid-2023. On the other hand, it is possible to see the “Air traffic” increasing from around 1,250 thousand to more than 2,000 thousand in the same period.

It is possible to see that these Maritime and Air transport modes have the same seasonality with a peak in the middle of the year. And arrivals and departures have very small differences.

# STATISTICS

## Descriptive Statistics

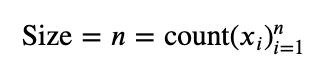
Descriptive statistics provide a numerical summary of the characteristics of a set of collected data. These calculations are designed to convey the central tendency, dispersion the shape of the distribution of the data set, among other characteristics.

Common descriptive statistics in general include:

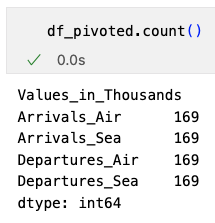
* Count or Size
* Minimum
* Maximum
* Sum
* Mean or Average
* Median
* Midrange
* Standard deviation
* Quartiles

## Count or size

The count or size of a dataset refers to the number of individual data points it contains. This is a measure of the dataset's magnitude in terms of its elements and is often denoted as *n* in statistical notation.

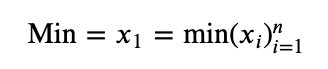


Pandas have the method “count()”, which returns the count of each column on the data frame and your type.

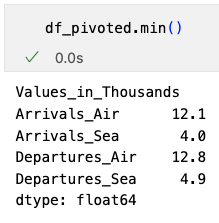


## Minimum

The minimum in a data set is the least value​, when the data is arranged in ascending order from the smallest to the largest value. Ordering a data set: “x1 ≤ x2 ≤ x3 ≤ ... ≤ xn”.

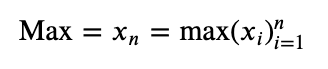


Pandas have the method “min()”, which returns the minimal value encountered in each column on the data frame and your type.

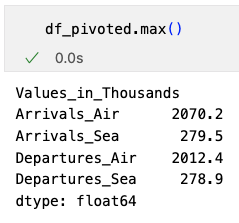


## Maximum

The maximum in a dataset represents the greatest value located at the far right when the data is ordered from the lowest to the highest value. Ordering a data set: “x1 ≤ x2 ≤ x3 ≤ ... ≤ xn”.

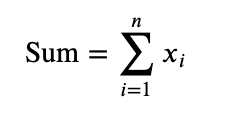


Pandas have the method “max()”, which returns the maximum value encountered in each column on the data frame and your type.

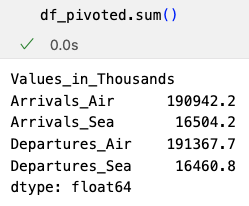


## Sum

The total of all data values in a dataset is known as the sum of the aggregate. It is calculated by adding together all the individual values in the dataset. This total is often symbolized by the Greek letter Sigma (Σ) followed by the expression for the data points, indicating the summation of the series of values: “x1 + x2 + x3 + ... + xn”.



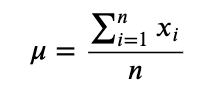
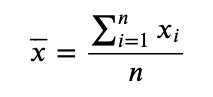
Pandas have the method “sum()”, which returns the sum value obtained from the sum of values from each column on the data frame and your type.



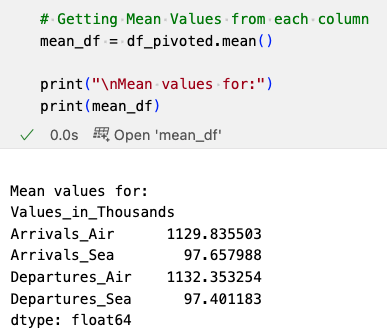
## Mean

The mean, also commonly referred to as the average, is a measure of the central tendency of a dataset. It is calculated by adding all the data values together to find the sum and then dividing this total by the number of data points in the set, which is the size or count.

For a Population: For a Sample:

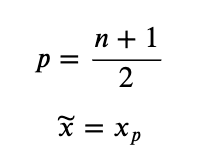
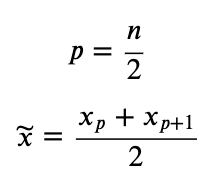
To collect this let’s use the method “mean()” from “Pandas”, creating a new data frame “mean\_df”, only with the mean values of each column.



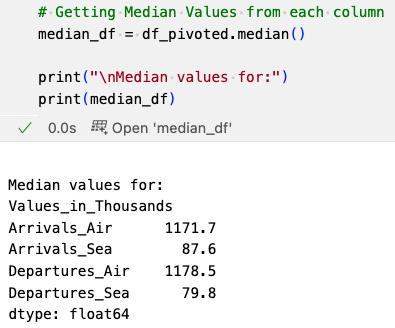
## Median

Median, when arranging a dataset from smallest to largest, the median is the middle value that divides the dataset into two halves. For datasets with an odd number of entries, the median is the central value. However, if the dataset has an even number of entries, the median is found by calculating the average of the two middle values.

If n is an odd number: If n is even the median is the average of the values at positions p and p + 1 where

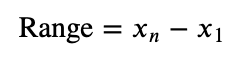
 

Obtaining these values for each column utilizing method “median()” from “Pandas”, creating a new data frame “median\_df”, only with the median values.

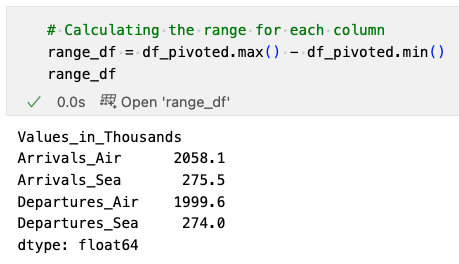


## Range

The difference between the minimum and maximum values in a dataset is known as the range. It is calculated by subtracting the minimum value, from the maximum value. The range provides a measure of the spread or dispersion of the data points within the set.

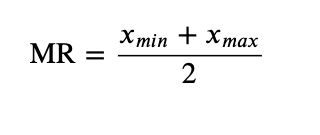


Pandas don’t have a straightforward method to return the range of the set but is easy to obtain just by subtracting the max value from the min in each column and storing it on a new data frame.

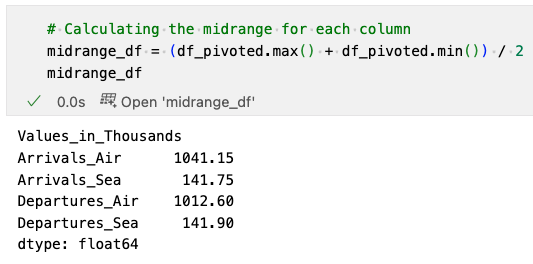


## Midrange (MR)

The midrange in statistics is found by averaging the smallest and largest numbers in a dataset. It provides a quick sense of the centre or middle value of the data, especially useful for understanding the data's range.



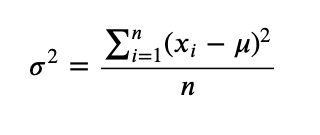
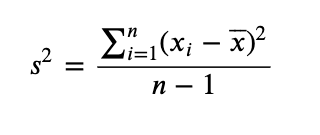
Pandas also, don’t have a straightforward method to return the midrange of the set but is easy to obtain just by subtracting the max value from the min in each column dividing the result by 2, and storing it on a new data frame.



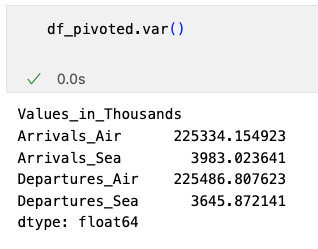
## Variance

Variance measures the spread of data points around the mean within a dataset. It's computed as the average of the squared deviations from the mean. A greater variance indicates that the data points are more dispersed from the mean, highlighting the variability within the dataset.

For a Population For a Sample

Obtaining these values for each column utilizing method “var()” from “Pandas”, creating a new data frame “var\_df”, only with the variance values.

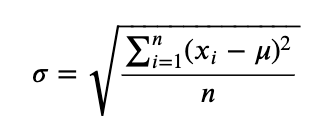
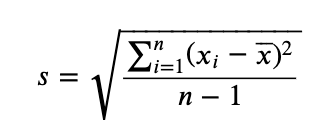


## Standard deviation

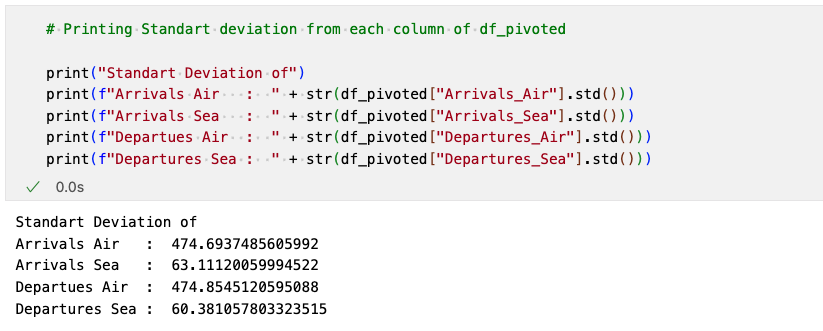
Standard deviation quantifies how spread out the data points in a set are from their average value. This measure is calculated by taking the square root of the average of the squared deviations of each data point from the mean. A larger standard deviation indicates that the data points are more widely dispersed from the mean.

The Standard deviation formula:

For a Population: For a Sample:

The library “Pandas” also has a function to make this calc for obtaining the standard deviation value, “std()”. Let’s perform this in each column from “df\_pivoted”.



## Quartiles

Quartiles divide a dataset into four equal parts.

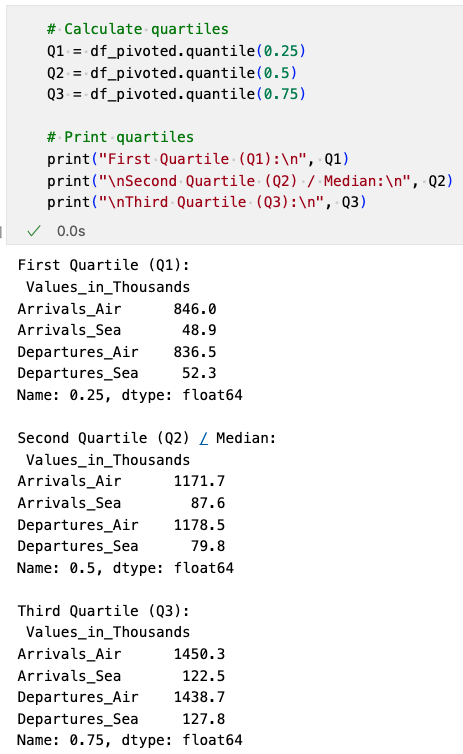
The median, or second quartile (Q2), splits the data into upper and lower halves.

The first quartile (Q1) is the median of the data points below Q2

The third quartile (Q3) is the median of the data points above Q2.

These quartiles help in understanding the distribution of data by highlighting the *spread* and *central tendency*.

The method “quartile()” on “Pandas” had to specify what quartile is to calc shown below.



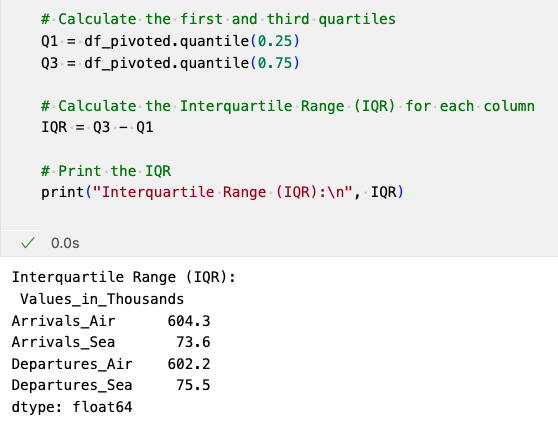
\**This is one of several methods for calculating quartiles.*

## Interquartile Range (IQR)

The interquartile range (IQR) is defined as the distance between the first quartile (Q1) and the third quartile (Q3) in a dataset. It represents the range within which the middle 50% of the data points lie, effectively measuring the spread of the central portion of the dataset and minimizing the impact of *outliers*.

𝐼𝑄𝑅=𝑄3−𝑄1

Pandas don’t have a straightforward method to do this but this is not hard to implement.



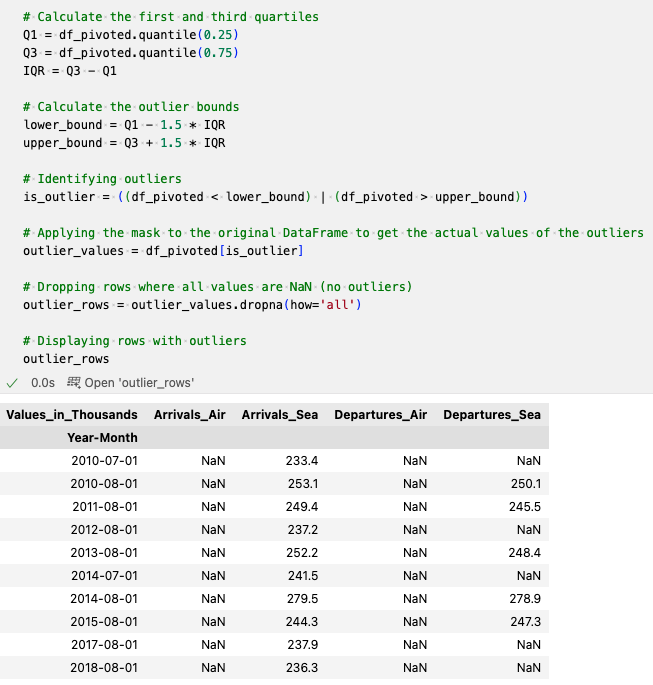
## Outliers

Potential outliers are those values in a dataset that fall either below the Lower Bound or above the Upper Bound. These bounds are determined by specific calculations that take into account the interquartile range (IQR), helping to identify data points that significantly differ from the rest of the dataset.

Upper Bound = 𝑄3+1.5×𝐼𝑄𝑅

Lower Bound = 𝑄1−1.5×𝐼𝑄𝑅

Below is the code how implement these formulas and detect the outliers.

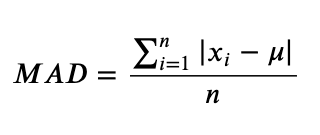
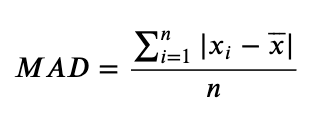


It is possible to visualize the result of this code with these parameters to specify the limits to detect outliers, 10 outliers were detected, all in the Sea sector.

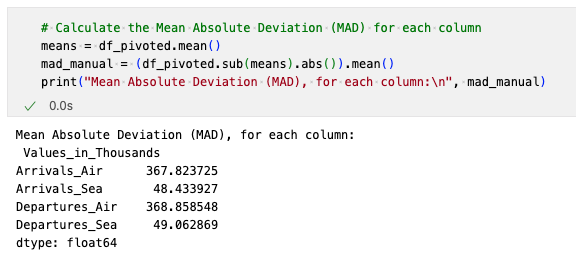
## Mean absolute deviation (MAD)

Mean absolute deviation (MAD), measures the average distance between each data point and the mean of the dataset. This is calculated by taking the absolute values of the differences between each data point and the dataset's mean, and then dividing by the number of data points. It provides insight into the variability of the dataset.

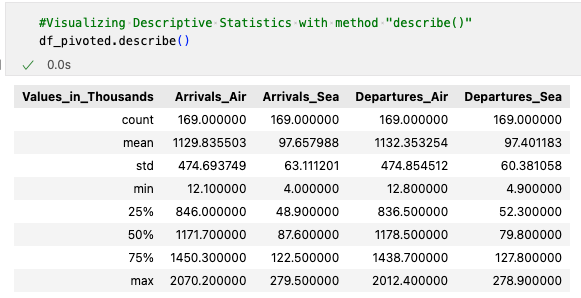
For a Population For a Sample

This subject has conflicts between some libraries when “mad()” in “NumPy” returns the absolute deviation from the median and in “Pandas” returns the mean absolute deviation so this “mad()” method was removed from some versions of the “Pandas” library. However, it is not difficult to implement manual code to perform this calculation as follows.



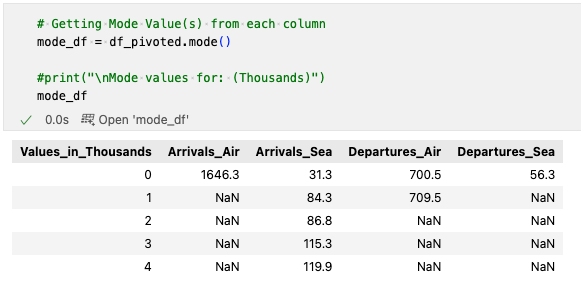
One method from “Pandas” can return a list of common descriptive statistics from each column from the data frame is “describe()”.



## Mode

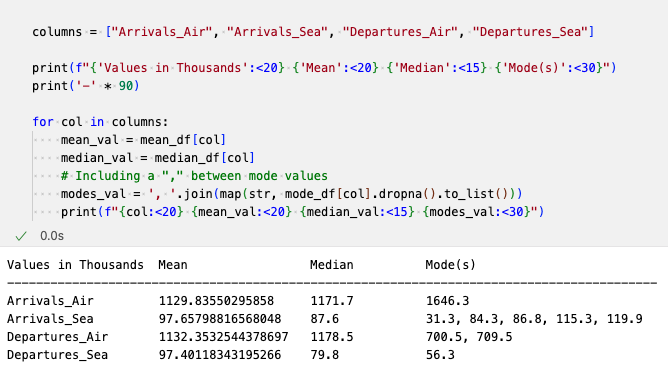
The mode in a dataset represents the value(s) that occur most frequently. When a dataset has only one mode, is called “unimodal”, when has two modes, called “bi-modal”, when has three modes is called “three-modal” and has more than three modes is called “multi-modal”. If multiple values occur with the same highest frequency, or if all values are unique then can call “no mode”.

Verifying these values from every column on the data set, utilizing the method “mode()” from “Pandas”, and creating a new data frame “mode\_df”.



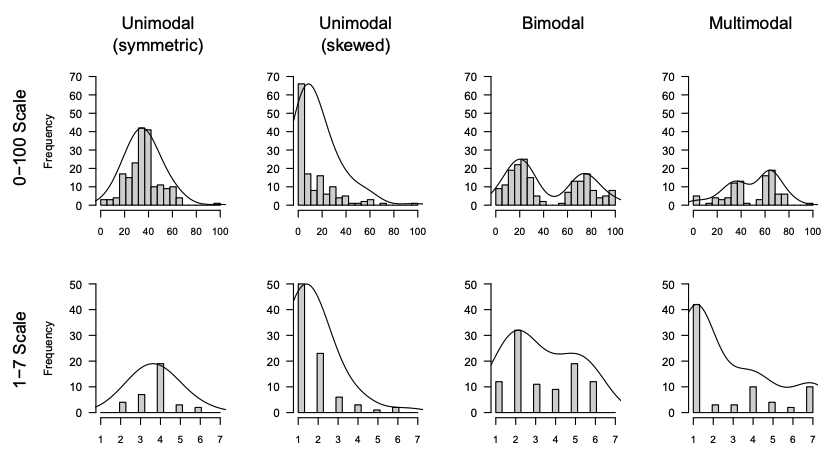
Observe the column “Arrivals\_Sea” has a list of modes, five in total, is a “multi-modal” and the column “Departures\_Air” has two “bi-modal” and “Arrival\_Air” and “Departures\_Sea” has only one mode each.

Let’s print the mean, median and mode, side by side from each column to better visualize. For this it is appropriate to implement a function with the method “for-loop”, concatenating in a tabular way with the method “print(f ‘{}’)”, the results to better visualize.



By analysing these three attributes it is possible to realize the dispersion of the data, “Skewness”.

Following is a picture showing visual differences between different types of models.



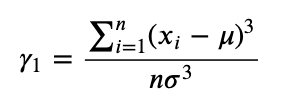
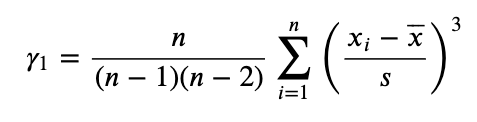
*Multimodality\_and\_Skewness\_in\_Emotion\_Time\_Series\_22Dec2022.pdf page. 04*

## Skewness

Skewness measures the asymmetry of a data distribution compared to the “normal distribution”. If the distribution has a longer tail on the left side, it is considered left-skewed or negatively skewed.

If it has a longer tail on the right side, it's right-skewed or positively skewed. This characteristic helps in understanding the direction and extent of distribution deviation from the symmetrical bell curve, (Hamilton, 2012)

For a Population: For a Sample:

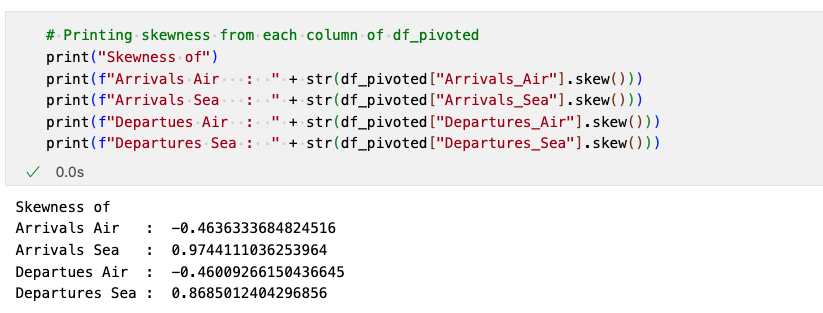
 

Examples of interpretation based on a relationship of mean, median and mode:

| **Relationship** | **Distribution Skewness** | **Interpretation** |
| --- | --- | --- |
| **Mean > Median > Mode** | Positively Skewed (Right-Skewed) | The distribution's tail is longer towards the higher values, suggesting a significant number of observations are larger than the central values. A few high-value outliers are pulling the mean to the right of the median and mode. |
| **Mean < Median < Mode** | Negatively Skewed (Left-Skewed) | The distribution's tail is longer towards the lower values, indicating a significant number of observations are smaller than the central values. Some low-value outliers are dragging the mean to the left of the median and mode. |
| **Mean = Median = Mode** | Symmetrical (No Skew) | The distribution is balanced on both sides, often indicative of a normal distribution. The values are evenly distributed around the center, with no skewness. |
| **­Mean = Median > Mode** | Moderately Positively Skewed | Similar to positively skewed, but the closeness of the mean and median suggests that while the distribution has a longer tail on the right, the skewness is not as pronounced. |
| **Mean = Median < Mode** | Moderately Negatively Skewed | Similar to negatively skewed, but the closeness of the mean and median indicates that while the distribution has a longer tail on the left, the skewness is mild. |

Then, it is possible to say that based on values obtained from column “Arrivals\_air”, mean < median < mode, your data distribution is “Negatively Skewed (Left-Skewed)”.

The library “Pandas” also has a function to make this calc, “skew()”. Let’s perform this in each column from “df\_pivoted”.



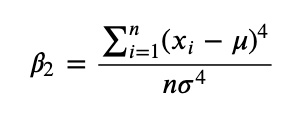
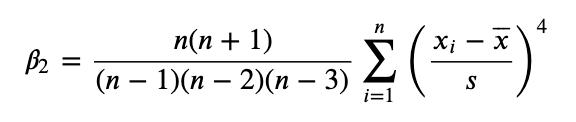
The result shows us the “Air sector” has a negative skewness, indicating it’s a left-skewed,

And the “Sea sector” has a positive skewness, indicating it’s right-skewed.

## Kurtosis ( Pearson Kurtosis )

Kurtosis measures the "tailedness" of a distribution, indicating how outlier-prone a dataset is. High kurtosis suggests more extreme outliers than a normal distribution, while low kurtosis indicates fewer extreme outliers. This helps assess the extremity and concentration of tail data compared to a normal bell curve.

For a Population For a Sample

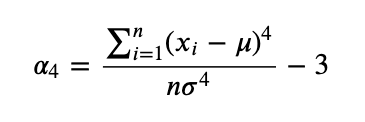
 

“Pandas” has a method that can return “fisher kurtosis or kurtosis excess” by default “kurtosis()”, and to find the “Pearson kurtosis” has to include an add of 3 on the calc. Let’s view the definition and formula of “Kurtosis Excess” to compare.

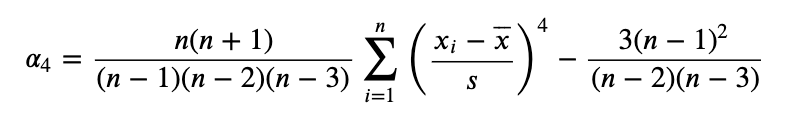
## Kurtosis Excess ( Fischer Kurtosis )

Excess kurtosis gauges the peak height of a distribution's tails, focusing on the concentration of outliers rather than their extremity. A distribution with high excess kurtosis indicates a significant presence of outlier data, pointing to more frequent extreme deviations from the mean compared to a normal distribution.

For a Population



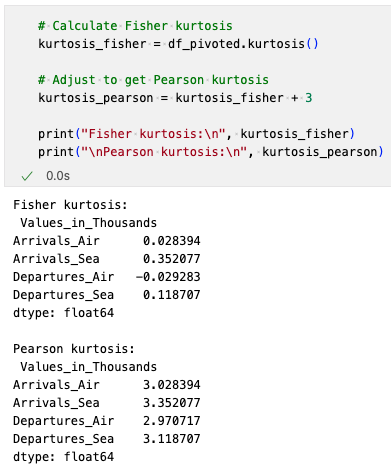
For a Sample



Excess kurtosis is calculated as the kurtosis of the distribution minus 3. This adjustment helps in identifying how the distribution differs from a normal distribution, where:

* A value of 0 indicates a shape similar to the normal distribution.
* Positive values indicate a distribution that is more peaked than a normal distribution (leptokurtic).
* Negative values indicate a distribution that is less peaked than a normal distribution (platykurtic).

Let’s implement the method “kurtosis()” over the “df\_pivoted”, data frame with the adjustment to Pearson Kurtosis.

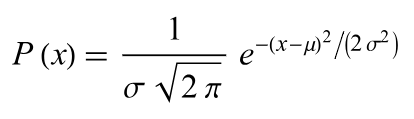
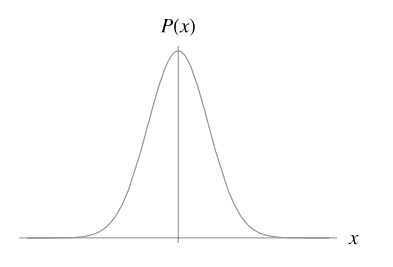


# NORMAL DISTRIBUTION / GAUSSIAN DISTRIBUTION

## Probability Density Function

While statisticians and mathematicians uniformly use the term "normal distribution" for this distribution, physicists sometimes call it a “Gaussian distribution” and because of its curved flaring shape, social scientists refer to it as the “bell curve.” Feller (1968).

A Probability Density Function (PDF) is a statistical expression that defines a probability distribution for a continuous random variable as opposed to a discrete random variable. The PDF describes the likelihood (probability) of the random variable taking on a given value. The most well-known example of a PDF is “Normal Distribution”.

*Picture from https://mathworld.wolfram.com/NormalDistribution.html*

## Kernel Density Estimate (KDE)

KDE is a non-parametric way to estimate the “probability density function (PDF)” of a random variable. It's smooth and not limited to a specific distribution shape (like normal or binomial distributions).

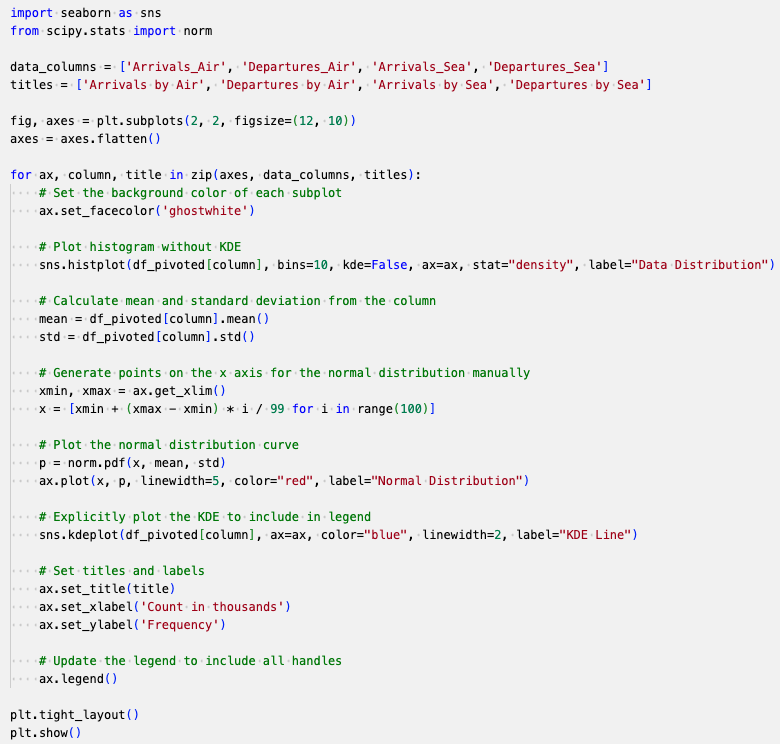
KDE works by placing a kernel (a smooth, bell-shaped curve) on each data point and then summing all these kernels to produce a smooth estimate of the data's density function.

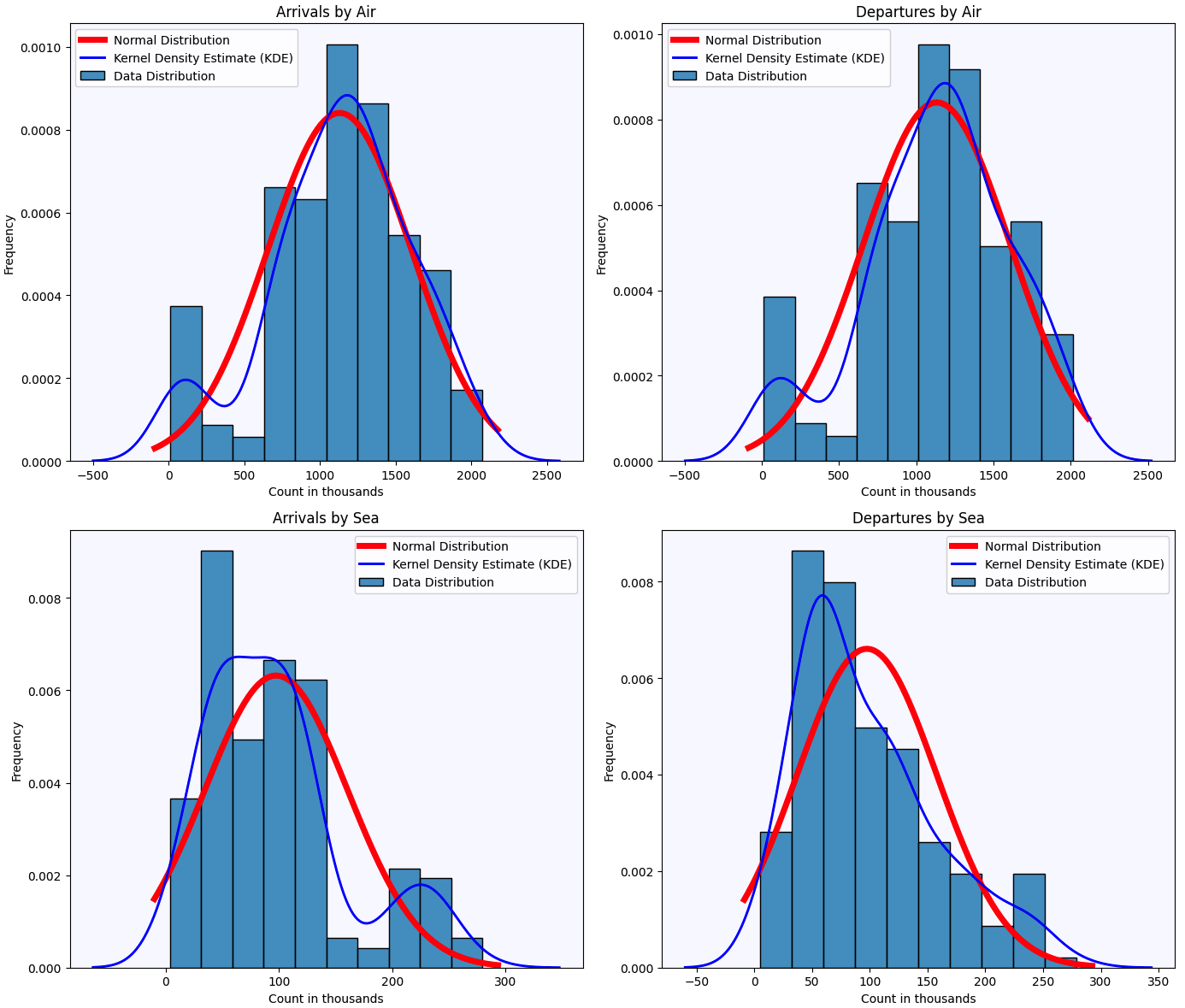
The final curve is a weighted sum of those kernels and gives a smoothed estimate of the dataset's density. (*Pandas.dataframe.plot.kde#*)

## Plot Comparative KDE x Normal Distribution

Now let's plot a separated histogram chart of each column of “df\_pivoted”, including lines to show “KDE” and “Normal Distribution” to compare the differences between them, and try to visualize, in consideration of the shapes of data distributions “skewness”, “kurtosis”, and “mode(s)”.

Considering “Pandas” and “MatPlotLib” have already been imported before, let’s import more two libraries “Seaborn” to make use of your method “kdeplot()” to trace and configure the parameters of the line “KDE” separately, and “SciPy” to make use of your method “norm.pdf()”, passing the values of mean and standard deviation to calculate values to plot the “Normal Distribution” line.





With these plots is possible to view the difference between the “Kernel Density Estimate (KDE)” line and the “Normal Distribution” line, when “KDE” still possible to view the strong influence of “Mode(s)” on the line, “Normal Distribution” smooth the line bringing more evident the “skewness” and “kurtosis” of the data set.

# 

# Difference between Discrete Data and Continuous Data.

## Definition of discrete data.

Discrete data refers to a form of quantitative information characterized by countable figures and non-fractional values. Typically, discrete data is presented in the form of whole numbers that convey precise quantities. A common way to conceptualize discrete data is to preface it with "the number of," for instance, the number of patrons in a shop. This kind of data generally encapsulates distinct occurrences that are already in the past. In analysing discrete data, you can examine precise numbers, such as the quantity of products sold on a particular date or the duration of time an employee has worked in a given week.

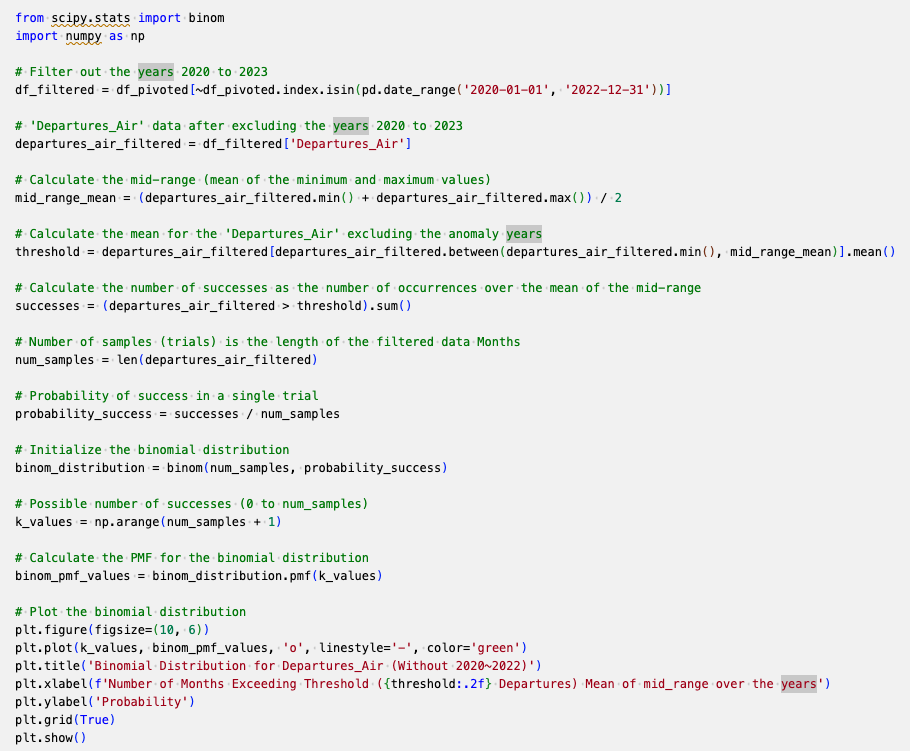
## Definition of continuous data.

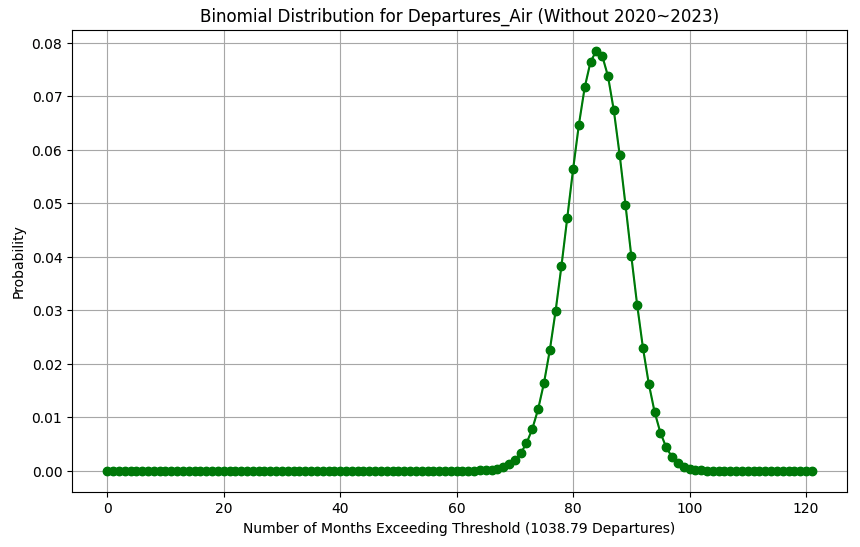
Continuous data is a quantitative data category that captures measurements that can be highly precise, extending to numerous decimal places as needed. It represents values that can be measured on a scale and can fall between any two amounts within a range. This data type is prevalent in sectors that demand exactness, such as healthcare, production, and research and development. Continuous data is dynamic, presenting the opportunity for organizations to scrutinize their processes and forecast upcoming patterns. An instance of its application could be a company monitoring the duration required by a team to fulfil assignments, providing insights into productivity and efficiency.

# BINOMIAL DISTRIBUTION

The binomial distribution is a discrete probability distribution that describes the number of successes in a sequence of independent experiments, where each experiment has two possible outcomes, often termed success and failure. The key conditions for a binomial scenario are that each trial is independent, the probability of success is identical for each trial, and there is a fixed number of trials.

Knowing this let’s try to plot a binominal curve from “df\_pivoted”, taking into consideration the years 2020 ~ 2022 it’s as anomalies and removing them from the data set to try getter better binominal probability curve. Also, let’s define success in this case months have the number of flights over the mean of the “midrange” over the years studied. Trying to set the success is the months when there are peaks of counts, representing high traffic. For this let’s use the libraries “Scipy.stats” with method “binom()”.



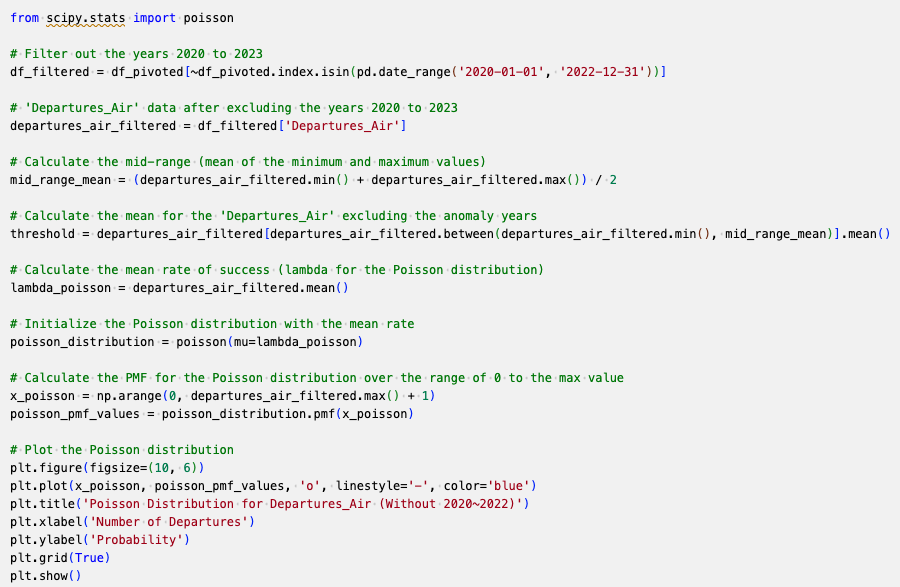


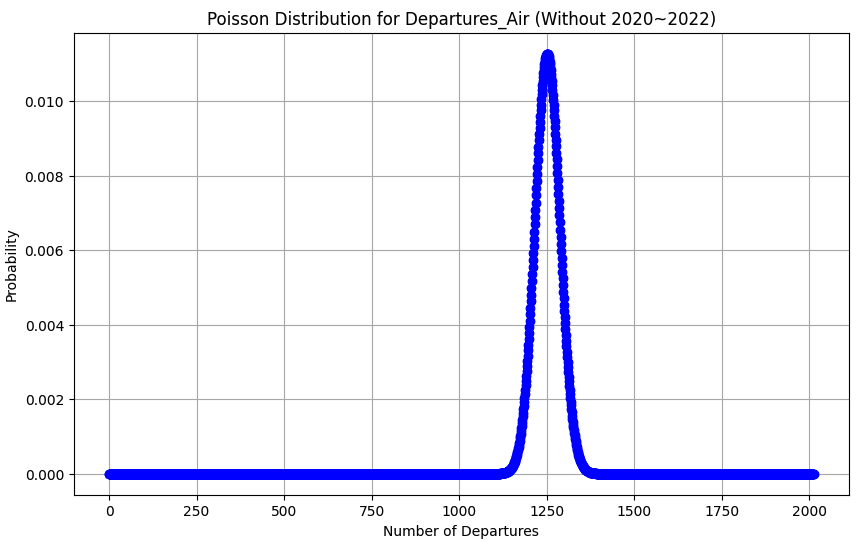
# POISSON DISTRIBUTION

The Poisson distribution is specified by one parameter: lambda (λ). This parameter equals the mean and variance. As lambda increases to sufficiently large values, the normal distribution (λ, λ) may be used to approximate the Poisson distribution.

Poisson distribution can describe the number of times an event occurs in a finite observation space. For example, a Poisson distribution can describe the number of flights on a specific month over the “mid-range” of the previous month. With probability equals a mean over the years.

Let’s try putting the same problem defined before on the code to draw the Poisson distribution line, but now specifying “lambda” as the success rate, with the same problem defined before, the number of flights over the mean of the “midrange” over the years studied, predicting with the same problem and the same parameters, the curve it is almost the same of “binomial distribution”. For this let’s now use the library from “Scipy.stats” with the method “poison()”.





# MACHINE LEARNING

## Project Management Framework

CRISP-DM is a robust and well-established framework that outlines a comprehensive process of understanding the business context, data understanding, data preparation, modelling, evaluation, and deployment. It's widely adopted due to its industry-agnostic and flexible nature. Because are the choice for this project. Make easier the study and development.

Supervised learning is the current case because the data set chosen has labels on data and predicts outcomes with good accuracy it’s the target. It is chosen when the goal is to model the relationship between measured features and an outcome.

GridSearchCV is a technique used to tune hyperparameters in machine learning models. It performs an exhaustive search over a specified parameter grid for a model and determines the best combination of parameters by cross-validation. Here's how it influences design decisions and problem-solving in a project:

Design Decisions:

Modularity: GridSearchCV encapsulates the hyperparameter tuning process, allowing for cleaner and more modular code. It abstracts away the intricacies of looping over parameters and can be used as a component within a larger machine learning pipeline.

Reproducibility: By using GridSearchCV, you ensure that the hyperparameter search is consistent and repeatable. This is crucial for scientific rigor and for sharing results with stakeholders or the research community.

Integration: It easily integrates with scikit-learn's machine learning models and can be used in conjunction with other cross-validation strategies and metrics.

Problem Solving:

Hyperparameter Optimization: Determining the optimal hyperparameters for models can be a challenging task. GridSearchCV automates this process and provides a systematic approach to exploring a wide range of hyperparameter combinations.

Performance Improvement: By thoroughly searching the hyperparameter space, GridSearchCV helps in finding the best model settings which can lead to improved performance.

Time Efficiency: While GridSearchCV may be computationally expensive, it runs multiple fits in parallel, significantly reducing the search time and making efficient use of available computational resources.

In the context of your project, you would use GridSearchCV as follows:

Define a hyperparameter grid that includes the parameters you want to tune and the values they should take.

Choose a scoring function that will determine how the models are evaluated.

Execute the grid search process, which will fit models with each combination of parameters in your grid, using cross-validation to evaluate each combination's performance.

Extract the best parameters and the best model to proceed with further model training or evaluation.

Here's an example code snippet illustrating the use of GridSearchCV:

# PROGRAMMING FOR DATA ANALYSIS

Python with Jupiter notebook bring the stat of art in facilities and improvements to work with data analysis. With a huge collections of libraries and methods scientific and statistical already implemented.

In the development of this data analysis project, I employed various programming paradigms that influenced design decisions and problem-solving approaches. Specifically, I leveraged the imperative, object-oriented, and functional programming paradigms to construct a robust and efficient analysis pipeline.

Imperative Programming: The project's foundation is built using an imperative style, which dictates the machine to perform sequences of operations to achieve a certain state. This was instrumental in data pre-processing and transformation, where step-by-step commands were executed to clean, normalize, and prepare the data for analysis. For instance, the data was looped through to handle missing values and outliers, ensuring that the subsequent analysis was based on quality data.

Functional Programming: Functional programming was pivotal for the statistical analysis and machine learning parts of the project. I utilized functions as first-class citizens to apply transformations to data sets in a stateless manner. Higher-order libraries and methods, such as **“Pandas”** and **“Scipy”**, were used extensively to manipulate collections of data and do scientific calculations. This paradigm particularly shone when applying the same operation across multiple data subsets, ensuring consistency and reducing the potential for errors.

Bottom of Form

# Reference list

Tufte, E.R. (1997) *The visual display of quantitative information / Edward R. Tufte*. Cheshire, CT: Graphics Press.

*Descriptive statistics* *from Wolfram MathWorld*. Available at: https://mathworld.wolfram.com/topics/DescriptiveStatistics.html (Accessed: 26 March 2024).

*Normal Distribution, Probability Density Function*, *The National Institute of Standards and Technology (NIST)*. Available at: https://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm (Accessed: 11 April 2024).

Aloorravi, S. (2024) ‘CHAPTER 4 Exploratory Analysis of Time Series Data’, in *Mastering Time Series Analysis and Forecasting with Python*. London, United Kingdom: Copyright © 2024 Orange Education Pvt Ltd, AVATM, pp. 142–142.

*Seaborn seaborn.heatmap - seaborn 0.13.2 documentation*. Available at: https://seaborn.pydata.org/generated/seaborn.heatmap.html (Accessed: 07 April 2024).

*Pandas* *pandas.DataFrame.corr - pandas 2.2.1 documentation*. Available at: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.corr.html (Accessed: 07 April 2024).

*Sklearn.preprocessing.MinMaxScaler*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (Accessed: 07 April 2024).

*Matplotlib.pyplot#* (no date) *matplotlib.pyplot - Matplotlib 3.5.3 documentation*. Available at: https://matplotlib.org/3.5.3/api/\_as\_gen/matplotlib.pyplot.html (Accessed: 07 April 2024).