Course: MSc in Data Analytics

Author: TEMUULEN Bulgan

E-mail: [2022427@student.cct.ie](mailto:%32%30%32%324%32%37@%73%74%75%64e%6e%74%2e%63%63%74%2e%69%65)

[temulenbd@gmail.com](mailto:temulenbd@gmail.com)

Student ID: 2022427

CCT College Dublin Continuous Assessment:

EXPLORATION AND ANALYSIS OF THE DATA:

“SUSPENSION OF PARKING BAYS IN DUBLIN CITY COUNCIL”

**Abstract**

I choose data on parking suspensions. It was published on the Transport section of the 'data.gov.ie' governmental website, a central portal with access to the governmental non-personal open data. The data I downloaded provides information about the on-street Parking Bay suspensions facilitating construction works, filming, broadcasting, loading, unloading, office moving, and many other events. There are 2092 rows and 8 columns of information in it covering the years between 2006 and 2011.

I did all the work by coding and performing calculations using Jupyter Notebook. And in this report, I'd discuss it in-depth, breaking it down into four sections with subsections

**Introduction**

Firstly, I would like to express my gratitude to the management team and the school's faculty team for allowing me to study and learn the Data Analytics Master's course in a healthy, supportive environment. It is convenient to have an online learning platform Moodle, an electronic library, and extra laptops to borrow when I need them. Secondly, this is my first time writing a report in a foreign language, so please be tolerant of my bad grammar and clarity. I appreciate your understanding and let's start introducing my assignment work.

Teachers instructed us to choose a particular data of interest from the governmental website of Ireland to process, analyse and interpret it. It took time to decide what data would suit my level of learning and understanding. Moreover, choosing a suitable rich dataset would help me identify possible problems at present and make predictions regarding the future.

After a week of hesitation, I decided and choose data on parking suspensions. It was published on the Transport section of the 'data.gov.ie' governmental website, a central portal with access to the governmental non-personal open data. The data I downloaded provides information about the on-street Parking Bay suspensions facilitating construction works, filming, broadcasting, loading, unloading, office moving, and many other events. There are 2092 rows and 8 columns of information in it covering the years between 2006 and 2011.

I did all the work by coding and performing calculations using Jupyter Notebook. And in this report, I'd discuss it in-depth, breaking it down into four sections with subsections.

1. **Creating the Data Frame and understanding of the Dataset.**

**(Jupyter notebook: 1 – 8)**

As I mentioned in my introduction, I spent a few days browsing all the datasets from the official government website data.gov.ie. The target of my assigned work is one of 26 different datasets labelled as "Transport and Infrastructure." I chose a data collection which was published by the Dublin City Council, and which contained the suspension information of on-street Parking Bays. I browsed a bit about the suspension rules in Ireland and what I learned from the Dublin City Council's official website was that by the request of the Roads and Streets Division of the City Council, on-street parking spaces can be suspended to provide room for filming, collection(setting-down), building work, office relocation and other events. The price depends on the location and duration, and it can vary from 12.70 to 50 euros for a single space per day. And my dataset included the fields of id, date of application, street, number of spaces to be suspended, the purpose of suspension, the date suspension commences, the date suspensions conclude, and the fee of charges.

The suspension file is in CSV format. After looking through the data in Excel, I created a new notebook with Jupyter and, as the first cell, imported all the libraries I'll use or might use for my analysis. Then I used Pandas' *read\_csv* method to create a new Data Frame called "df" with the *filepath\_or\_buffer* argument <dccparkingbaysuspensionsp20110930-1044.csv'>. However, it raised a <UnicodeDecodeError>, and I couldn't read the file. So, I imported Universal Encoding Detector Python library to detect the character encoding for the text. I ran the file, scanned the whole data, and printed the results. The encoding for my file was <ISO-8859-1>, but I used <latin1> format for the reading because it accepts any possible byte as input.

I successfully created a Data Frame, so I started examining it to understand its characteristics and the information it held. I checked the data's structure using the *head* and *tail* methods. I used the *dtypes* method to see the data type, and I found that with the exception of the column <Id>, it solely contained object data. With the *info* method, I learned that there are some missing values in 6 columns. The column <Date> and <Locations of Spaces> has one missing value, two for the <Purpose> column, four in <Date Suspended>, and <No of spaces>, <Date Expired> has five, seven respectively.

Following the examination, I was aware that there were a few crucial tasks I had to complete before beginning the analysis. First off, I have to conduct a lot of cleaning, grouping, and encoding work on the column with the categorical data because it makes up the majority of my data collection. There are also a few DateTime format columns; I must choose the correct time format to use for them and decide whether to keep all of them or not. For numerical columns with an object type, I must transform them to the correct value types. And lastly, I'll have to handle each column's missing value.

I detailed all this data cleaning and manipulation work in the following part.

1. **Data cleaning and manipulation.**

**(Jupyter notebook: 9 – 101)**

Before moving on to the main analysis, I prepared and validated my data in this section. This part is divided into five separate subsections that collectively cover the entire cleaning and modification procedure. And I just wanted to mention that since data cleansing is my favourite part of the entire analysis, I did a lot of experimenting and spent the majority of my time on it.

1. Renaming columns, adding new columns, and re-arranging them*.*

(Jupter notebook: 9 – 10)

I began this section by removing the <Id> column and adding the columns <str class> and <duration>. The <Id> column is removed because it doesn't include any data that can be used for analysis, and doing so will make it simpler for me to concentrate on the other crucial components of the Data Frame. In the next subsection of this section, you will find explanations for creating the <str class> and <duration> columns.

The next step was to use the *for loop* and rename each column. I changed the name strings to all lowercase letters and substituted spaces between words with underscores because it makes calling functions and using the column names much easier. Along with that process, I also rearranged the position of the columns.

1. Data type conversion of each column.

(Jupyter notebook: 11 – 16)

When I examined the data in the first section, I learned that only column <Id> has numerical data and since we removed this column now all the columns have object values. I must take a close look at them all. This sub-section is all about the datatype conversion and I’ll do the conversion for each column separately depending on the type of data.

*location of spaces (Jupyter notebook: 11)*

For the <location of spaces> column's data type, I created a custom function to pair with the *apply* method. This custom function examines each value of the column individually. It will either return an empty string as a value or the current value of the string, depending on whether the supplied condition is true. When I applied this function, I also checked the data type of the column. Since this column contains only the names of the streets, there is no need for any other further adjustments.

*purpose (Jupyter notebook: 12)*

The column < purpose> contained the reason for the suspension of a parking bay. So, I did the same procedure for this column. Created a custom function to pair with the *apply* method. Then after examining each column's value individually, replaced it with an empty string or the current value's string. And because there is no need for any other adjustments, I left it as it is.

*date, date\_suspended, date\_expired (Jupyter notebook: 13)*

<date>, <date suspended>, and <date expired> columns all have object data types despite being supposed to have DateTime format. I changed the data type to the appropriate one.

*number of spaces (Jupyter notebook: 14 – 15)*

When we examined the Data Frame info, an object type was returned even though the field <number of spaces> is meant to have integer values. I used the value counts method to determine the cause of this and discovered from the results that the column contained some user inputs that were a combination of numbers and letters. There are six values with the mix, and I changed them with the right value using the *replace* function after I switched the data type to numeric because the column contains the continuing value.

*amount paid (Jupter notebook: 16)*

I created a custom function for the <paid amount > column to be used with the *apply* method. The function replaces extra symbols rather than numbers by scanning each value individually. And after it is checked if the value is equal to zero or not. When the value is equal to zero, it is replaced with <Nan>, or else with the numbers.

1. Accessing, modifying, and handling the values.

(Jupyter notebook: 17 – 46)

In this subsection, I focused just on the values of the entire Data Frame. I checked the data, eliminated any duplicates, and then reset the indexes. Then carefully examined the values in each column independently.

*location of spaces (Jupyter notebook: 23 – 36)*

For column <location of spaces>, using the original values would be pretty complicated. The column values were entered manually by random users, resulting in many linguistic and spelling errors. Moving forward, using them would be very difficult, and I cannot simply delete them from the Data Frame. My best approach was to find other raw data with the street names and replace the erroneous values with the standard terms. I browsed online and found a CSV file containing typical road and street names in the Transport section of the government's “data.gov.ie” website. A few columns in it capture road and street names by street class, area code, area type and sub-zone.

Using the new CSV file with the common street names, I created a new Data Frame called <df standard names>. It has 4771 rows and 13 columns in total. The only column I need now is the column with standard street names. So, I scanned with the *for loop* all the values from the <df standard names>'s <street name> column and did the same to the main Data Frame's column <location of spaces>. After that, I compared the values using the *in* operator and replaced user inputs with common street names. I run backwards and forwards between the two Data Frames twice. Now, since the *for loop* found and replaced the primary Data Frame values, I could distinguish correct values from erroneous ones by the case of the first-string letter.

When the *for loop* did its job, I checked all the unique values containing the upper-case first-string letter and found in total 655 of them. I used many options to scan faulty values and replace them with the standard names, but unfortunately, I failed all attempts. I had tried Sequence Matcher's string prediction, Fuzzy Wuzzy's processing, for loop, while loop, regular expression, etc. I found that I must do everything manually if I face values with the user's input error. And that's what I did. I created two list variables, one with half-strings of the faulty cell and another with common names and using *for loop*, I found and replaced them all.

After numerous time-consuming, long-lasting looping operations, I could clean the column without deleting or changing any of the values. However, it's not the happy end for this column. Without grouping or classifying it, using it in the Data Frame would be difficult. It has numerous street locations throughout Dublin which doesn’t make sense. I then started seeking solutions and ideas online. According to the information provided on the official governmental website, prices vary according to the parking zones. There are five zones with different colors: yellow, red, green, orange, and blue. They are defining the location of the streets. If I found a file with these classifications, I could easily classify the streets into their colored zone. But unfortunately, I was able to find only one CSV file that contained 100 rows of locations and zone groupings which is not enough for my street list. I then decided to only use the data from the <df standard names> Data Frame.

*str class (Jupyter notebook: 37 – 46)*

In the first subsection of this section, I mentioned adding new columns to the data set. The <df standard names> Data Frame has a few columns with information on the street class including code, new area code, subarea, and subarea EA. I checked all these values on Google Maps and decided to use only the <street class code> value for the street classification. For the new column <str class> I used the values from the <location of spaces> column. I wrote code with loops and filled this column with the corresponding value. The main target of the code is to scan the street name columns on both Data Frames and make a comparison. If the string value existed on both columns, it must insert the related value of the street class code into the new column. After *for loop* performed its operation, I checked the unique values for the column <str class> and found five empty and four values with the corresponding number “1.5”. To replace the empty values, I used the *to replace* method; for the values with street class code “1.5”, I used the mode value of the column.

1. Missing values of the Data Frame.

(Jupyter notebook: 47 – 94)

In this section, I continued the value-handling process, but I handled only the missing values for each column this time. I started from the column <amount paid>.

*amount paid (Jupyter notebook: 47 – 52)*

From the previous sub-section, I know that there are non-values in this column and wanted to check the reason for the no payment. When I printed all the purposes for the non-payment, I saw that it was mostly public-related government-dependent activities such as blood donations, marathons, fundraising, exhibitions, festivals, ceremonies, the restoration of state-owned buildings, etc. Simply put, the government won't pay for the activities that they organize themselves. So, the best thing I can do here is just simply to remove the rows with zero payment.

I also looked for values below zero because the payment for the field <amount paid> cannot be a negative integer. Only one row, with a value of “-76.0” euros, was found. After carefully analyzing the row and applying logic, I concluded that it is yet another user input error.

*duration (Jupyter notebook: 53 – 65)*

The length of the parking bay suspension as stated on the official government website is measured only in days. Using the input values from the <date expired> and <date suspended> columns, I can easily determine the value for the new <duration> column. Once again using the *for loop*, I wrote some code and verified the values of the two date columns. When the data were equal to zero, I simply entered “None” in the duration column value. However, if they were DateTime formatted inputs, I pulled them out and calculated the number of days that I needed to insert into the "duration" column.

The code worked and now when the column has the values inserted, I have to check the non-values. I printed out all the rows which contained the empty input and started analysing it. The pay amount of suspension for rows 155 and 203 is 1400. The <date suspended> and <date\_expired> columns are blank. However, the <date> column contains starting date. In this situation, the only thing I can do is assume the duration value using calculations. By dividing the total amount spent by the mean value of the current parking suspension pricing rates, I can calculate the approximate number of days. I filled empty sections with the calculated values, and I may also require <date suspended> and <date expired> fields in the future. I used the value of the <date> column as the suspension date and for the expiration date, I used the sum of the total duration days with the <date> value.

I applied the same method to rows 339 and 389. 2100 has been paid for each suspension, I calculated the mean and divided the paid value with the calculated number. And on these rows, neither the < date suspended > column nor the < date expired > column included any information I am aware of. So, I again used the value of <date> and <duration> columns to calculate the missing values. Identical method for row 1430.

The <duration> column doesn’t have any missing values, but I need to check the values to ensure there are no faults. I ran the value counts method and found four rows with higher numbers containing incorrect data in them. Rows with the index numbers 6, 301, 1508, and 1512 had user input issues. After replacing them with the correct duration, I ran again the value counts method for negative numbers. The result is four rows with the same issue: rows 194, 376, 630, and 1478.

*no of spaces (Jupyter notebook: 66 – 69)*

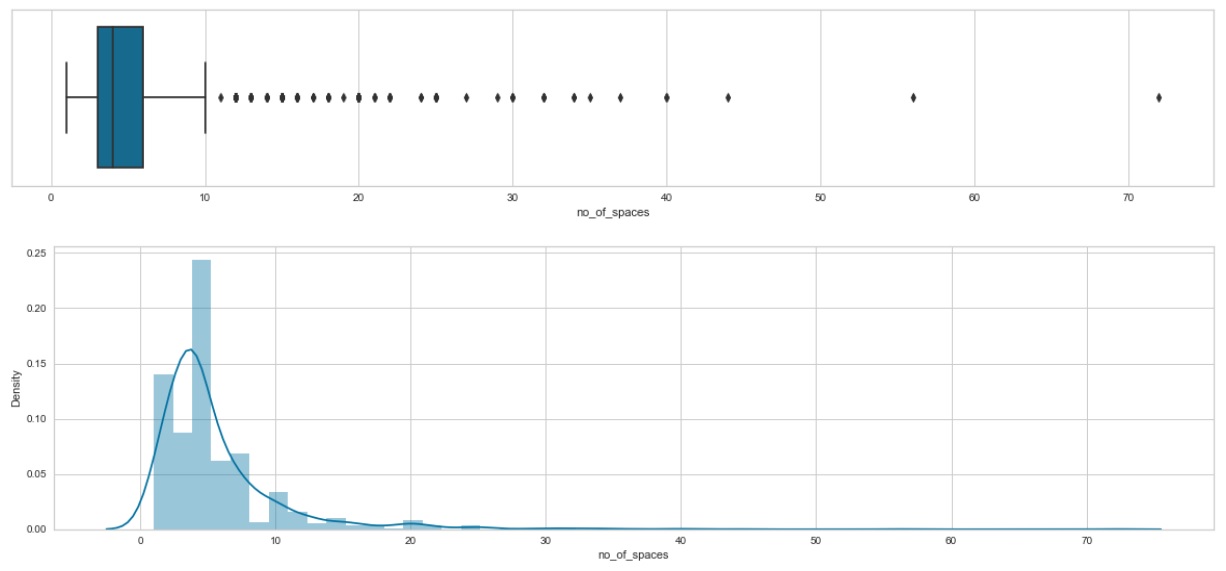
I found three non-values in the column <no of spaces> by using the isnull.sum method. I plotted the distribution boxplot and distribution plot for observation because I couldn't ignore the missing values in this column. The column values have a right-skewed distribution, as seen in Figure 1. This indicates that the mean and median are not equal in this instance, therefore it is recommended to think of using the median value to fill in the missing values when the data is skewed. I'm replacing the non-values with the median value for this column, which I calculated to be "4".

Figure 1. Distribution boxplot and distribution plot of the < no of spaces> column.

*purpose (Jupyter notebook: 70 – 80)*

Information on the temporary suspension of parking spots is on the Dublin City Council website shows that filming, delivery (set-down and collection), building work, office relocation, and broadcasting units are the main reasons for the suspension. From the unique values of the "purpose" column, I can see that these values are unorganized and difficult to use.

First, I looked at the values in the <purpose> column that related to the objectives of the filming, I found 665 rows of them. Despite the word "filming" being included in the string value, there were many differences between each string.

After a hesitation, I decided to create a custom function, catch each purpose value, and then substitute it with one of the five essential purposes. I started with the filming purposes and found 733 matters related to it. Then 97 values for office relocation, 236 for broadcasting, 120 for building works, and 220 for loading and unloading.

The primary five purposes had finally been grouped. But when I checked the values that were left from the classification, I found out that there are more than 300 of them are not related to any of the main five. They were connected to celebrations, roadhogs, carnivals, charitable activities, displays, etc. I printed out the information and read each of them; there was no other way to name them as others.

*purpose (Jupyter notebook: 81 – 88)*

In my future calculation, I’m not going to use the column date; for this reason, I renamed the column into the <year>. And because I’m using only years here, I removed the day and month information from it. After these changes, I checked the unique values and found one containing the wrong information. Printed out the whole row for that analysed and replaced it with the correct one.

I printed all the unique values again and found out that there were only 85 entries for the year 2011. I will use the years for statistical analysis; if it is not complete information, it won’t give me a reasonable interpretation for that year. So, I decided to remove all rows which included the year 2011 in it.

*date\_suspended (Jupyter notebook: 89 – 94)*

When I calculated the values for the <duration> column, I checked all the months and days, and it can’t have errors. That’s why I used the lambda function to extract only the year values from there, checked it and found three different year values displayed with only one

count. The year 2005 and 2001, and 2011 are incorrect, and I replaced them with the correct ones. I need the value of days from this column, so I’m using another lambda function to extract it. Okay after checking with the *head* function everything seems all right.

1. Final adjustments to the columns

(Jupyter notebook: 95 – 100)

The values of the columns are now organized and prepared for analysis. But before the analysis, I’m deleting the columns that I won’t be using in the future. These columns are <date expired> and <location of spaces>. I deleted the <date expired> column because I already added a column called ‘duration’ by using its information. And the information from <location of spaces> column is already extracted and used for the column <str class>.

After all the cleaning and data manipulation processes, I now have data set with 1657 rows and eight columns.

**3. Statistical analysis (Jupyter notebook: 101 – 124)**

1. Descriptive statistics

(Jupyter notebook: 101 – 118)

Descriptive statistics describe, show, and summarize the basic features of a given dataset. It can help me understand the data better. In this section, I can only represent the available data of the sample and can't include theories, inferences, probabilities, or conclusions. There are four main categories of descriptive statistics, and I'll be using three of them for this assignment - measures of frequency, central tendency, and dispersion or variation.

*3.1.1. Measures of frequency (Jupyter notebook: 101-106)*

Let's begin descriptive statistics with the frequency measure. The frequency of an occurrence in statistics is the number of times the observation occurred during an experiment or research. These frequencies are often shown graphically or in a table. I'll use frequency distribution for my dataset's nominal and ordinal columns.

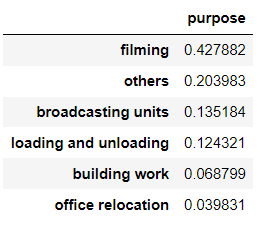
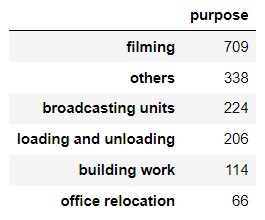
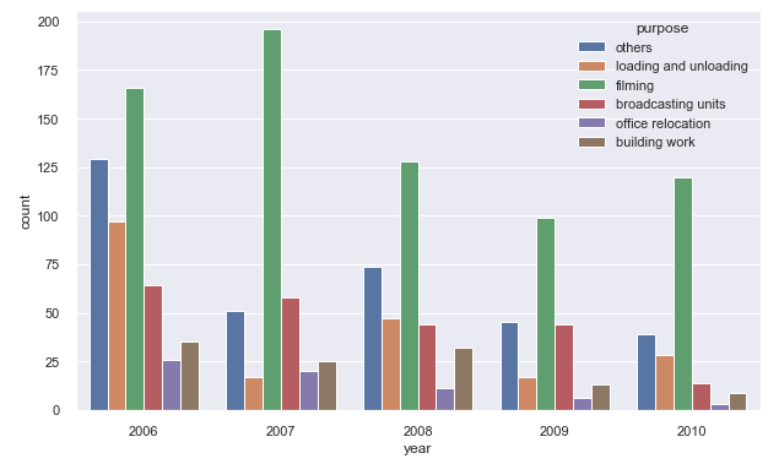


Figure 2. The percentage of values for Figure 3. The count of the values for

the <purpose> column. the <purpose> column

If I group my data according to the reason for the suspension, I can see from the *Figure 2* and the *Figure 3* that most suspensions were temporarily suspended for filming. A total of 709 suspensions between 2006 and 2010 were made for filming purposes, representing nearly 43% of suspense during this time. Office relocation occurs for 4% of all postponements and is the lowest cause of parking suspension.

**

The *Figure 4* demonstrates that, between 2006 and 2010, the most common reason for suspensions was filming. Whereas the number of office relocation is the lowest each year. Additionally, the counts for the rest of the purposes vary uniquely annually.

Figure 4. The number of counts of the column <purpose>

buy the years.

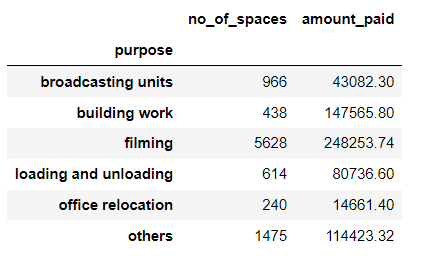
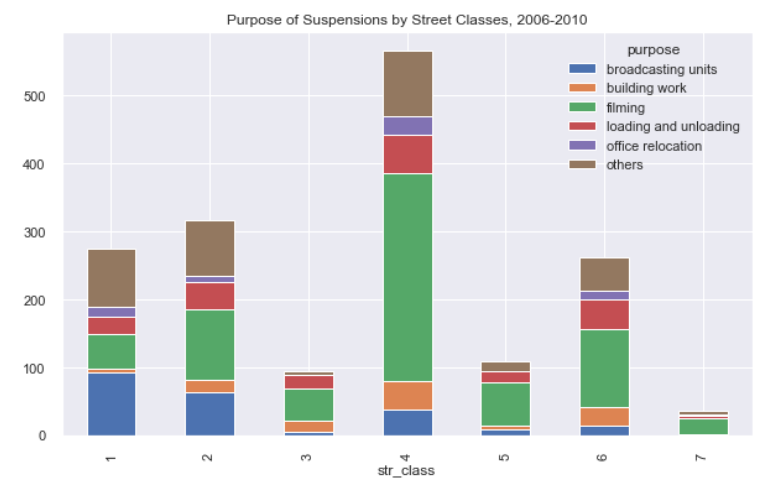
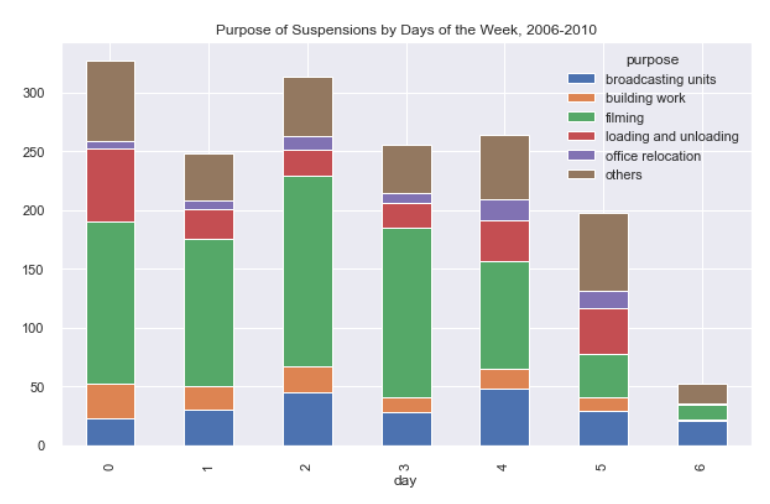
The *Figure 5* shows that the filming has the highest numbers in both categories when we count the number of spaces and the number of earnings for each purpose. Once again, office relocation has the lowest counts for both. One interesting observation is that while building work suspension uses relatively few parking spaces, the amount of money the government makes from it is far more than the average for other uses.

Figure 5. The count of the columns <no of spaces>

and <amount paid> by its purpose.



The stacked graph "Purpose of Suspensions by Street Classes" shows that the 4th class street is where most suspensions took place and that this is also where filming occurs most frequently overall. Even though the 7th street class had the lowest count of all suspense, filming in this column is the highest by the numbers. An interesting observation from this graph is that the 1st street class is the only place where the filming count outnumbered the number of broadcasting and other suspensions.

The second graph illustrates the number of suspensions occurring by day of the week. Here I can see that the government has the highest request for suspense during the weekday, and Sunday has the lowest. During the weekdays, filming suspense takes the lead by the numbers, but there is a different image on weekends. On Sunday, the most significant digit is for other purposes, and on Saturday, broadcasting and other purposes have higher amounts for a parking suspension.

*3.1.1. Measures of central tendency (Jupyter notebook: 101-106)*

A measure of central tendency is a summary measure describing a whole set of data with a single value representing the middle or center of its distribution. There are three main measures of central tendency: the mode, the median and the mean. Each of these measures describes a different indication of the typical or central value in the distribution.

In this part of the assignment, I want to see the central values of the continuous variables <number of spaces>, <duration>, and <amount paid>. I'll find the means, medians, and modes of the previously mentioned columns. The code for the calculation is in a Jupyter notebook.

Mean value for:

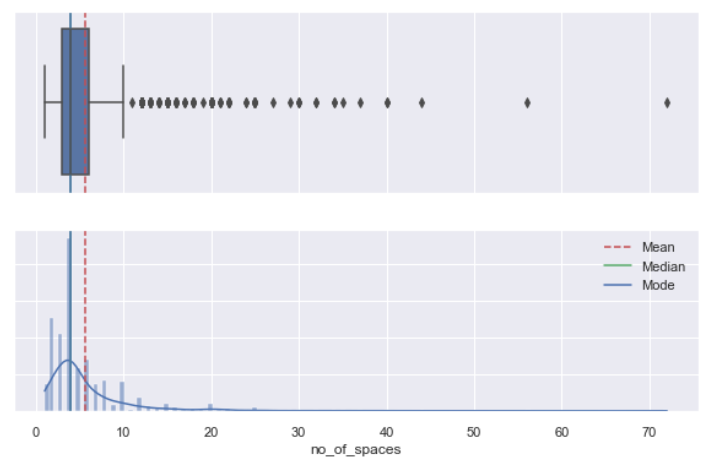
* the column <no of spaces>: 5.649366
* the column <duration>: 3.118286
* the column <amount paid>: 391.504623

Median value for:

* the column <no of spaces>: 4.0
* the column <duration>: 1.0
* the column <amount paid>: 190.0

Median value for:

* the column <no of spaces>: 4.0
* the column <duration>: 1.0
* the column <amount paid>: 152.0

All three columns have a right-skewed distribution based on the data computed above. Since the mean of all three figures is greater than the median and the mode, I am confident in that statement. As an example, I'll create a boxplot and a bar graph of central values for one column.

*3.1.2. Measures of central tendency (Jupyter notebook: 111-118)*

Every data consists of some variability within its range. Variability in data is defined as how far apart the data lie from each other and the centre of the distribution. It is also known as the shatter, spread or dispersion of data. Dispersion of data is defined as the degree to which the arithmetical value is approached to spread an average value. The measure of dispersion helps in calculating the variability of data. Like the central tendency of data, variability is also essential for summarizing data characteristics. It helps in stating the facts and figures of the data. I'll calculate the values of the dispersion measures and try to summarize them

***Range*** *(Jupyter notebook: 111)*

The data spread between the lowest and highest value in the distribution is known as the range. Let's find out those values for the columns that have continuous data.

* the column <no of spaces>: 71.0
* the column <duration>: 485.0
* the column <amount paid>: 17185.0

The range is quite a helpful indication of how spread out the data is, but it has some severe limitations. It's because sometimes data can have outliers that are widely off the other data points. In these cases, the range might not give a true indication of the spread of data for the column <duration>. Due to its outliers, the column <duration> has a very high range value.

***IQR*** *(Jupyter notebook: 112)*

The interquartile range gives us the range of the middle half of a data set. It also provides a general understanding of the values' distribution and degree of clustering. I will calculate the IQR for the columns with continuous values and print the results below.

* the column <no of spaces>: 3.0
* the column <duration>: 1.0
* the column <amount paid>: 232.6

The high value for the column <amount paid> indicates that the central portion of my data is spread out further. Conversely, smaller values for the column <no of spaces> and the column <duration> show that the middle values cluster more tightly.

*Variance (Jupyter notebook: 113)*

The term variance refers to a statistical measurement of the spread between numbers in a data set. More specifically, variance measures how far each number in the collection is far from the mean (average) and, thus, from every other number in the set. I will calculate the variance for the columns with continuous values and print the results below.

* the column <no of spaces>: 27.38
* the column <duration>: 209.56
* the column <amount paid>: 761917.99

The more spread the data, the larger the variance is in relation to the mean. In comparison to the variation of the column <amount paid>, the variance of the columns <duration> and <no of spaces> is relatively low. It indicates that the <amount paid> column has a widespread.

*Standard deviation (Jupyter notebook: 114)*

A standard deviation is a measure of how dispersed the data is in relation to the mean. A low standard deviation means data are clustered around the mean, and a high standard deviation indicates data are more spread out. I will calculate the variance for the columns with continuous values and print the results below.

* the column <no of spaces>: 5.23
* the column <duration>: 14.47
* the column <amount paid>: 872.88

CONCLUSION *(Jupyter notebook: 115)*

From the output of the *explain* method, I can see that the columns <no of spaces>, <duration> and <amount paid> with continuous data have a right-skewed distribution. That is a reflection of the column's median and mean values. Although the IQR value for the column's "duration" and "amount paid" is low, the range between the minimum and maximum values for the mentioned columns is significant. It indicates that there are outliers in the columns. The column "amount paid" has a relatively high standard deviation, suggesting values spread widely across the distribution.

1. **Machine learning.**

**(Jupyter notebook: 125 – 148)**

For the machine learning section, I used CRISP-DM

I broke the process of data collection into six major phases:

* Business Understanding.
* Data Understanding.
* Data Preparation.
* Modeling.
* Evaluation.
* Deployment.

I didn’t have time to finish the machine learning report in word, so please understand it and find my machine learning models in my Jupyter notebook. Thank you

References:

Witte, R.S. and Witte, J.S. (2019) *Statistics*. John Wiley & Sons. Canada

Burkov, A. (2019) *The hundred-page machine learning book*. Andriy Burkov. Canada

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. (2009) *The elements of statistical learning*. Springer. United States of America

David freedman. (2007) *Statistics*. W. W. Norton & Company. United States of America.

Matthew Kirk. (2017) *Thoughtful machine learning with Python*. O’Reilly Media, Inc., United States of America

Andrew Wolf. (2022) *Machine learning simplified*. Themlsbook. Online

Mark Summerfield. (2010) *Programming in Python 3*. Addison-Wesley Professional. United States of America

Wes McKinney. (2017) *Python for Data Analysis*. O’Reilly Media Inc. United States of America

Chun-houh Chen, Wolfgang Härdle, and Antony Unwin. (2008) *Handbook of Data Visualization*. Springer. United States of America

*Suspension of parking bays in DCC - DCC\_PARKINGBAYSUSPENSIONS\_P20110930-1044.CSV* *(2021) Data. Gov.IE*. Available at:

https://data.gov.ie/dataset/suspension-of-parking-bays-in-dublin-city-council-area/resource/b3d97186-f6ff-4160-aa63-8762cc4d11ad (Accessed: October 10, 2022).

*Roads and streets in DCC - Dublin City Roads and streets CSV* *(2016)* *Data.Gov.IE*. Available at:

https://data.gov.ie/dataset/dublin-city-roads-and-streets/resource/294c5bf2-c89c-4c59-955d-d66f10ca428b (Accessed: October 10, 2022).

Learn. Scikit. Available at:

https://scikit-learn.org/stable/index.html (Accessed: November 11, 2022)

Main Page. Wikipedia. Available at:

https://en.wikipedia.org/wiki/Main\_Page (Accessed: November 11, 2022)