Text

Description automatically generated

Jader Leonardo Gomes da Rosa – **sba22199**

**sba22199@student.cct.ie**

**Suspension of Parking bays in DCC**

**Continuous Assessment 1**

**Dublin**

**November 11,2022**

**TABLE OF CONTENT**

|  |  |
| --- | --- |
| INTRODUCTION | 3 |
| Data Manipulation | 4 |
| Exploratory Data Analysis | 5 |
| Statistics and appropriate plots | 9 |
| Boxplot payment per year | 9 |
| Add a calculated column on Data Frame | 11 |
| Created new Data Frame | 12 |
| Report the payment | 12 |
| Boxplot number of spaces per year | 15 |
| Number of spaces per amount in 2006 | 17 |
| Distribution (Binomial) | 18 |
| Decision Trees and Random Forest | 19 |
| Decision TreeRandom forest | 20 |
| Explain about SEMMA | 23 |
| Explain about CRISP-DM | 23 |
| Problems with Testing Machine Learning Models | 25 |
| Principles & Best Practices | 25 |
| Reproducibility | 26 |
| How to Test Machine Learning Models | 26 |
| Invariance Test | 27 |
| Matplotlib pyplot: Pair plot | 28 |
| Columns description and details | 31 |
| CONCLUSION | 35 |
| REFERENCES | 31 |

**INTRODUCTION**

I am working with the database of suspension parking bays in DCC, the data contains the number of spaces and values by period, so it was possible to work with report’s graphic generation by periods and map the amount of space.

I identify some inconsistent on CSV document and I corrected as necessary, the inconsistencies and their due correction were indicated in the first topic of this document.  
All information contained in this document was based on instructions obtained in the classroom and all complements are correct referenced.

**Data Manipulation**

Adjustments were made to the database to obtain correct data.

1 - Created a column "C" to inform only the year.

2 - created a column "B" to inform only the month and year.

3 - The future period was identified where the data was run to the correct year. (07/06/2006 instead of 07/06/2026).

4 – The row with no data was removed.

Some “location\_of\_spaces” in the column “E” were adjusted because we identified some similar ones with different abbreviations or double space.

1 – The “146 Parnel St. Dublin 1”, “146 Parnell Street” and “146 Parnell Street. Dn 1” were changed to “146 Parnel Street Dublin 1”

2 – The “17, Shops, New Cabra Road, Cabra Dublin 7”, “17 Shops New Cabra Road Dublin 7” and 17 Shops, New Cabra Road were changed to “17, Shops, New Cabra Road, Cabra Dublin 7”

3 – The “Adelaide Road (in front of No. 24)”, “Adelaide Road, Dublin 2” and “Adelaide Road” were changed to “Adelaide Road, Dublin 2”

4 – The “Amiens Street, Dublin 1”, “Amien Street (Opposite Connelly Station)”, “Amiens Street”, “Amien Street”, “Amiens Street opposite Luas” were changed to “Amiens Street, Dublin 1”

5 – The “Anglesea Road (Outside No. 7)”, “Anglesea Road”, “Anglesea Road (outside the Herbert Park Hotel)” were changed to “Anglesea Road”

6 – The “Aungier Street, Redmond's Hill” and “Aungier Street, Redmonds Hill” were changed to “Aungier Street, Redmond's Hill”

7 – The “Baggott Street Upper” and “Baggot Street Upper” were changed to “Baggot Street Upper”

8 – The “Blackhall Place” and “Blackhall Place” were changed to “Blackhall Place”

9 – The “Castle Street” and “Castle Street” were changed to “Castle Street”

10 – The “Dame Court” and “Dame Court, Dublin 2” were changed to “Dame Court, Dublin 2”

11 – The “Georges Quay” and George's Quay” were changed to “George's Quay”

12 – The “Glengarriff Parade” and “Glengariff Parade” were changed to “Glengariff Parade”

13 – The “Grenville Street” and “Greenville Street” were changed to “Greenville Street”

**Exploratory Data Analysis**

Some libraries were imported to execute the codes. During the project’s process, I had needed to add more libraries emerged, which were made according to it needed and in order of need. The main ones were loaded in the first line of Jupyter Notebook

Exploratory data Analysis (EDA) is a technique composed of a range of code that allows an analysis of the consistency and possible issues in the database.

There are two types of analysis,

**Univariate:**

Univariate will give you the summary of one variable, you can use this method to get the summary of a variable.

**Bivariate:**

Bivariate will give you the summary of two variable, you can use this method to get the summary of two variable.

If you want to get the summary of one variable, you can use univariate method, if you want to get the summary of two variable, you use bivariate method. If you want to get the summary of more than two variable, you use multivariate method.

The following commands were applied to EDA.

Print out with “head” the first 5 rows that include zero.

Graphical user interface, text

Description automatically generated

With this command we can identify that there is following column order by “Id”, “month\_year”, “year”, “date”, “location\_of\_spaces”, “no\_of\_spaces”, “purpose”, “date\_suspended”,”date\_expired” and “amount\_paid”.

With that, I was able to improve the database and insert a small adjustments that made my database more comprehensive to be able to generate other models.  
  
Basic information, this informative command shows how many columns there are in the database, what is the name of each column, number of items filled in for each “Non-Null” column, the type of data contained in each column and the size of file.

Graphical user interface, application

Description automatically generated

With this, it was possible to identify which columns could generate graphs and

statistics for a possible approach and introduction of Machine learning (ML).

The amount paid column (being float) combined with column “year” we can generate annual’s graphs or monthly payments.

With the number of spaces columns (being float) we can generate statistic with number of spaces in each row of the database.

The shape is simply the command that shows how many rows (2091) and columns (10) there are.

describe()

This function returns the count, mean, standard deviation, minimum and maximum values and the quantile values for the given series.

count()

This function returns the number of items in a series, excluding NaN values.

mean()

This function returns the mean of the values for the requested axis.

std()

This function returns the standard deviation of the values for the requested axis.

min()

This function returns the minimum value of the given series.

max()

This function returns the maximum value of the given series.

quantile()

This function returns values at the given quantile.

Graphical user interface, text, application

Description automatically generated

The “shape” command returns quantitative information for the numbers/quantity of columns and rows are there in the Data Frame (DF)



Applying a Filter to return a table with the filters done, this is a great command to generate grouping graphs and show results by specific periods (this can be done with any column). In this case I performed a filter for 2007.

Graphical user interface, text, application

Description automatically generated

The command “duplicated” with “print” we can see if there are any duplicated rows. 

EDA is essential for every project start, the database must show reliability, solidity and all the data must have a plausible coherence so that it is possible to execute the commands and show the correct results without discrepancies, deviations or even errors due to their inconsistency.

EDA provides an overview of the database and what can be worked and performed.

**Statistics and Appropriate plots**

**Boxplot payment per year**

The Boxplot applied to the column of “amount\_paid”, we can identify the highest payment insider.

Graphical user interface, table

Description automatically generated

In the next results we can see the filter command applied along with the Boxplot and we can see results by year.  
  
Graphical user interface, table

Description automatically generated

The 2008, 2009 and 2010 results, with an emphasis on the “2009” results, where we have a low number of values between zero and 500.

Graphical user interface, table

Description automatically generated

The last report in boxplot per year ends in 2011, in which case we can see a low amount of incidence a non-compliance with the other graphs and low payment values.

Graphical user interface, application, table

Description automatically generated

**Add a calculated column on Data Frame**

Based on the Dublin City website, it was possible to identify the size of a parking unit, with this base I added a new column on df to calculate the amount of space it has.

Graphical user interface, text, application, email

Description automatically generated

**Created new Data Frame**

Creating a small df called “size\_df” to specify it this size of the parking suspension will be considered small or large and everything that estimates between them will be considered medium.

Graphical user interface, text, application

Description automatically generated

**Report the payment**

In this graph we see the total payments by year.

Chart, bar chart

Description automatically generated

I applied a data separation and created a new data frame (df) for the years 2006 and 2007 separated by monthly.

Table

Description automatically generated

The following graph shows the total paid per month for 2006 and 2007 per month.

We can say that in the months of November for those 2 year they had a relationship and a lower payment than October for same years.

The largest amounts of payment were in January, March 2006 and September, December 2007.

Chart, bar chart

Description automatically generated

If we want to introduce the year that we want to visualize and show the graph by month for specific year, an input was introduced inside the commands to generate the results based on the imputed value. (E.g., If we input 2008).

The largest amounts of payment were in October and May 2008.

Chart, bar chart

Description automatically generated

E.g., we input 2010

The largest amounts of payment were in September and December 2010.

Chart, bar chart

Description automatically generated

**Boxplot number of spaces per year**

The next demonstrations we can identify the number of spaces by year and their representativeness and greater insider for 2006.

Graphical user interface, application, Word

Description automatically generated

Other boxplot for number of spaces per year

A picture containing graphical user interface

Description automatically generated

Again, we identified a difference between 2010 and 2011 and it is relationship with the amounts paid for the same years.

Chart

Description automatically generated

**Number of spaces per amount**

the graphs we can see the highest (top 10) incidence of number of spaces for a specific year (2006) defined by input.

In 2006 the incidence of 4 spaces is greater than 200 times, different from the suspension of payment with 1 space that in 2006 had only 25 incidences

Chart

Description automatically generated

**Distribution (Binomial)**

Poisson and binomial form scipy.stats were imported to Jupyter Notebook to perform probability calculations for exact values.

Tests were made to generate an exact probability for an input calculated based on the representative “number of spaces” of input in the total on database and counting it is frequency within the database. Graphical user interface, text, application, email

Description automatically generated

Inside the app “f(x).app” I put the information obtained by Jupyter Notebook I can verify that the probability of a parking suspension is equal to 5 spaces is 0.18099 (obtained value exactly equal I have seen on Jupyter Notebook that was calculated.

Chart, histogram

Description automatically generated

The variance of a binomial distribution.

It can be used for statistical testing, for estimating population parameters, or for making predictions about future events. For example, a researcher may use binomial distribution to test a hypothesis about the probability of a certain event occurring. The researcher may also use binomial distribution to estimate the population mean or the population variance. For example, if the expected value for a binomial distribution is 5 and there are 10 trials, the probability of observing 5 or more successes is 0.67. Using Binomial Distribution Binomial distribution can be used in a variety of ways. One common use is in hypothesis testing, where the null hypothesis is that the probability of success is equal to some known value and the alternative hypothesis is that the probability of success is not equal to the known value. Another common use for binomial distribution is in calculating confidence intervals. For example, if you wanted to know the 95% confidence interval for the probability of success for a binomial distribution with 10 trials and an expected value of 5, you would calculate the 95% confidence interval as 5 +/- 1.96. Binomial distribution can also be used to calculate the probability of observing a certain number of successes in a given number of trials. For example, if the expected value

**Decision Trees and Random Forest**

Having tested the commands for decision trees and random forest, the following demonstrations present the results obtained in the Jupyter Notebook

I delimited a size and classification the size by number

Small = 0 , medium = 1 and large =2.  
  
Graphical user interface, application

Description automatically generated

**Decision TreeRandom forest**

Graphical user interface, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface

Description automatically generated

Text

Description automatically generated

Decision TreeRandom Forest It is easy to learn and implement. It has high accuracy as compared to other models.

This is also a decision tree-based model. It consists of many decision trees combined. It is more accurate as compared to a single decision tree. It carries a high accuracy.

1. Logistic Regression

- Logistic regression is a statistical technique used for binary classification. It is used to predict the probability of an event occurring. The output is between 0 and 1. It is easy to interpret and implement.

2. Linear regression

- Linear regression is a statistical technique used for predicting continuous values. It can find a relationship between two variables or more. The output is continuous. It is easy to interpret and implement.

3. Support Vector Machines

- Support vector machines are the type of supervised machine learning algorithm could be for classification and regression problems. It is a versatile algorithm that can be used for linear or nonlinear problems. It is easy to interpret and implement.

4. Neural networks

Neural networks are the type of machine learning algorithm could be for classification and regression problems. It is a powerful algorithm that can be used for linear or nonlinear problems. It is difficult to interpret and implement.

Decision trees:

Advantages

* Decision trees are easy to use, as they require little data preparation.
* The decision tree can handle both categorical and numerical data.
* The decision tree is not biased by outliers and is not affected by them.

Disadvantages

* The decision tree is prone to overfitting.
* Decision trees can create complex trees that do not generalize well to the data.
* The decision tree may not be accurate if the data is biased.

Random Forest

Random forest is the supervised learning algorithm that could be for classification and regression problems. The Random Forest algorithm is an ensemble learning method that is built on decision trees. It builds multiple decision trees and takes the average of all the predictions.

This algorithm could be for both classification and regression problems. It is one of the most accurate Machine Learning algorithms. It is used for data that is structured and unstructured. The Random Forest algorithm is not biased by outliers.

**Explain about SEMMA**

SEMMA is a very detailed process; however, it is not widely used because it requires that the user have some knowledge of data mining and statistics. The knowledge required to use the process, as well as the software, can be acquired through training courses. However, training courses are not widely available to all users of the SAS software.

**Explain about CRISP-DM**

CRISP-DM, which stands for Cross-Industry Standard Process for Data Mining, is a testament to how the industry is steering its data mining efforts.

As a methodology, the following are descriptions of the typical phases of a project, the tasks involved in each step and their explanation of the relationships between these steps.

As a process model, CRISP-DM provides an overview of the data mining lifecycle.

1. Data Understanding

The first phase in identifying the lifecycle of a data mining project is understanding the data. This phase is important because it establishes the context for future data mining efforts.

During this phase, data miners must answer important questions such as:

What business problem am I trying to solve?

What kind of data do I have?

Where does this data come from?

How can I best represent this data for data mining?

What are the important features in this data?

What are the potential benefits of data mining?

What are the potential risks?

2. Data Preparation

The second phase of the data mining life cycle is data preparation. This phase is important because it ensures that the data is ready for data mining.

During this phase, data miners must answer important questions such as:

What data do I need for data mining?

How can I best represent this data for data mining?

How can I cleanse and transform this data?

3. Data Mining

The third phase of the data mining life cycle is data mining. This phase is important because it is where the actual data mining takes place.

During this step, we have to see that it is still important to understand some of the issues raised during these later steps for long-term planning and future data mining goals.

**Problems with Testing Machine Learning Models**

**1. Testing.**

Testing is about confirming whether our model works as expected, whereas evaluation is about assessing the model’s performance.

You may not be able to test a machine learning model in the traditional sense.

These two concepts can be very different in machine learning.

You may not be able to test a machine learning model in the traditional sense.

If you train a model on a training set and measure the accuracy of the predictions on the same data, the model will have a very high accuracy.

But if you evaluate the model on a different set, the accuracy will be much lower.

To properly understand the performance of your machine learning model, you need to measure it on a holdout set.

You can’t test a machine learning model because you can’t control the behaviour of the model.

You can only evaluate the performance of the model and iterate if you’re not satisfied with the performance.

To test a machine learning model, you need to generate test data from the same distribution as the training data.

With traditional software, this is not a problem because the input and output of the software is well-defined and you can generate test data

**Principles & Best Practices**

Unit testing is essential for code that will be reused and for code that is part of a computationally intensive pipeline.

Integration testing is essential for code that will be deployed on a platform or as a web service.

Model testing is essential for code that uses Machine Learning.

A/B testing is essential for code that will be used in a production system.

A/B testing can be used to compare different models, different data sets, or different parameter settings.

Regression testing is essential for code that is part of a long-running process or that is subject to change.

Test coverage is the percentage of code that is covered by tests.

The goal of testing is to find bugs, not to prove that the code is correct.

Testing is an essential part of the development process, but it is not a substitute for good design.

**Reproducibility**

1. Ensure that all components run in the same order and receive the same random seed.
2. Generate the data in your experiments from a known distribution.
3. Generate the parameters of your models from a known distribution.
4. Document what versions of libraries were used.
5. Document how the experiments were run.
6. Make sure that the data, code and experiments are stored in a version control system (e.g. git)
7. Make sure that the experiments can be reproduced from scratch.
8. Make sure that you can reproduce the plots in your paper.

The goal is that anybody should be able to reproduce the results in your paper.

**Data**

1. Make sure that your data is stored in the version control system.
2. Make sure that your data is stored in a machine-readable format (e.g. csv).
3. Make sure that you store the raw data and not the derived data.
4. Make sure that you keep track of which files were generated from which files.

**How to Test Machine Learning Models**

Here are some tips to get started:

1. Define your testing process and objectives upfront
2. Break your data into separate training, validation and test sets
3. Train your model on the training set and evaluate it on the validation set
4. Tune your model’s hyperparameters until it performs well on the validation set
5. Once you’re satisfied with your model, evaluate it on the test set to get an estimate of its true performance

**Applying their logic on Machine Learning behaviour**

There are two types of tests in machine learning: model accuracy testing and model performance testing.

**Model accuracy testing:**

1. Unit testing: Check that each individual component in your machine learning pipeline works as expected.
2. Integration test: it is the verification step to see if the different commands work correctly and are in machine learning set.
3. End-to-end testing: Check that your machine learning pipeline works as a whole, from data pre-processing to model training to model deployment.

**Model performance testing:**

1. A/B testing: Compare the performance of two different machine learning models on a test dataset.
2. Cross-validation: Evaluate the performance of a machine learning model by training it on different subsets of the training data and testing it on the remaining data.
3. Hyperparameter tuning: Find the best values for the hyperparameters of a machine learning model by training the model on different combinations of hyperparameter values and testing it on a validation set.

**Invariance Test**

This test is essential to ensure that the models we build can generalize to new data. This is especially important when we are building models to predict outcomes for new individuals, new groups of individuals, or at new times.

Invariance test is a good way to check whether our assumptions about the model are valid. It helps to assess whether the model is robust to changes in input data. By testing the model with different inputs, we can verify that the model behaves as expected.

There are many ways to test invariance. One common method is to randomly permute the input data and see if the model output changes. If the model output is unchanged, then the model is invariant to the permutation of the input data.

Another method is to add noise to the input data and see if the model output changes. If the model output is unchanged, then the model is robust to noise in the input data.

Testing invariance is important because it helps us to understand the limitations of the models we build. It also helps us to understand how the model will behave when we deploy it in the real world.

**Matplotlib pyplot: Pair plot**  
This type of graph shows various information within the database and its comparison with other various in the form of graphs, an input was added to classify this demonstration by year

Chart, box and whisker chart

Description automatically generated  
Detailing some of these graphics we can see for example:

The “amount\_paid” vs “no\_of\_spaces” graph. we see that your data is accumulated between spaces smaller than 20. but we also have exceptions like the case of a data point that is almost centred at 5000 in “amount\_paid” and 40 in “no\_of\_spaces”.

Another interesting piece of information is how the data for the "array" vs "m2" or "no\_of\_spaces" is correctly separated, the classification made was very accurate and with these graphs you can see its accuracy.  
The type of classification is to delimit the "hue" by "m2" thus showing a graph with more details for all periods

As in this chart, we detail the m2 table by colour, the lightest being the smallest data points and the darkest being the largest.

Calendar

Description automatically generated with medium confidence

were separated by 5 lines, the lightest being up to 300 m2, then 600 m2, 900 m2, 1200 m2, the darkest being 1500 m2.  
Table

Description automatically generated

Continuing with the analysis in the same parameters of the previous chart, we can detail more information, such as:

The same graph of "amount\_paid" vs "no\_of\_spaces", we see a single isolated data point in the range close to zero in "no\_of\_spaces" and above 15000 in "amount\_paid" and rating as 300m2 or less.

This points to an irregularity because supposedly the smaller the space, the lower the payment amount, but we see that exactly at this data point there was a divergence that may or may not be an error.



interesting point of information is the behaviour of the graph of "year" vs "array". there are exactly 3 points for each year per array with only one greater than 1500 m2 in 2009 with array 2.

Calendar

Description automatically generated

Analyzing the graph of "no\_of\_spaces" vs "year", I can say that for all years, their greatest representation is for cases with less than 25 spaces and, consequently, they are smaller than 600 m2 as it is fully related to this value.

we also see similarity with the relationship of "amount\_paid" vs "year", we can say that the largest payment insider is less than 5000 for all years, but we can see some points that showing values ​​of "m2" greater than 600m2 and 900m2. And the inverse we can also see, payment values ​​close to or greater than 10000 for the years of "maybe" 2006, 2008 and 2010 with m2 less than or equal to 300m2.

A picture containing diagram

Description automatically generated

**Columns description and details**

Graphical user interface, application, Word

Description automatically generated

at times it was necessary to use the "dtype" command to identify the type of information contained in each column, for example:

0. column "Id" all its values ​​are sorted with "Int64", that means with their values ​​are absolute in decimal places.

1. column "month\_year" (Note: this column was created directly in the database, treating its classified date as objects, and generating another object classification but with only the month and year to which it refers) with Object, this means that the values ​​are text, dates or a mix between them.

2. column "year" all its values ​​are sorted with "Int64".

3. Column "date" containing the days, month, and year to which that line refers.

4. column "location\_of\_spaces", containing the addresses of the spaces (Note: some data were run to maintain a standardization in the address, because small errors were identified, such as double spacing or abbreviations for the same address).

5. column "no\_of\_spaces" all its values ​​are classified as "float64" it means that its values ​​are number containing decimal places.

6. column "purpose" the data is classified as "object". This column contains different text with many company names or just localities where it was not possible to use it to develop other reports, graphs, data, or ML based on this column.

7. column "date\_suspended" this column because data of the day, month and year that the suspension occurred, this column is classified as "Object"

8. "date\_expired" column is classified as "object" containing day, month and year.

9. column "amount\_paid" all data in this column are classified as "float64", containing the amounts paid for each parking suspension launched in df, this was very important information for the project, which allowed for several analyses, comparison, and groupings.

10. column "m2" all data classified as "float64", this column was added during the project to give more coverage and develop other lines, this information was created based on information on the Dublin government website and the calculated of variable with its relation linked directly with the space number to delimit its total size by m2.

11. column "size" was classified as "object" containing a size classification (small, medium, or large) depending on its size per m2, this data was supposed and possible modification depending on the project needs. I specified that all values ​​less than and equal to 100m2 would be classified as small, values ​​greater than 100m2, less than and equal to 200m2 would be classified as medium, and all values ​​greater than 200m2 would be classified as large.

12. column "array" was classified as "Int32" this means that its values ​​are absolute and with a small range of several between them. This column was created due to the need for a numerical classification for small, medium, and large types, this numerical classification was generated as follows: all small classifications are classified with "0" in the array column, medium classifications are classified with "1" in the array column, and large sorts, are sorted with "2" in the array column.

Another type of code that displays information is "info ()".

This command numbers the columns (as I detailed in the previous item), informs the amount of information contained in each column, for example:

Data Frame contains 2072 padded lines for most columns, except for "purpose" which contains 2071, "date\_suspended" which contains 2069 and for "date\_expired" which contains 2066 padded lines.

in addition, it also informs the size of the file (in this case the file is 283.1+ KB) and the number of columns that contain that type of data, for example:

Df contains 3 as "float64", 1 as "int32", 2 as "int64" and 7 as "object"

Graphical user interface, text, application

Description automatically generated

**CONCLUSION**

We can conclude that the database has a solidity for analysis with data grouped by price and generate report and graph for years.

Other analysis made was based on the number of spaces and their total footage.

Several reports were generated.

For the bar graph, it was possible to group the amounts paid and separate them by year, showing the entire amount for the years between 2006 and 2011.

Another bar chart had the same format as the previous one (total amounts paid per month) between January and December of each year or grouping two years together.

The report was not made for a month with all the years from 2006 to 2011 because the chart would be long with little space for display and its visualization would not be attractive.

Boxplot reports were made to identify possible discrepancies or gaps between the consolidated database with every year from 2006 to 2011 and divided by years for better visualization and identification of control points.

This boxplot was made for the total amount paid for all years or divided year by year, the same sense was applied to the number of vacancies all for all years or divided year by year.

Another bar chart was also made to identify the amount of the number of spaces and the amount of insider for a given period defined by an input.

The probability graph with binomial was generated from the application and its data were placed as generated in the Jupyter notebook, this made the generated report have its accuracy tested and its result is the same as the command "binom.pmf"

The supplementary data came from a public source in Ireland and with them I was able to determine the total value per m2 based on your amount of parking space.

Analyzing the database, we see that there are payments made with amount "0" to 150 and these amounts are repeated, and their amounts paid are different depending on their location and purpose.

Based on this information I decided to take a more accurate approach and multiply the amount of space by the m2 of a parking space by the number of spaces.

Another additional information was the creation of a df to characterize the size of the space between small, medium, and large and we can classify this characteristic with a number, with zero being equal to small, one equal to medium and two equals to large.

**REFERENCES**

https://www.digitalocean.com/community/tutorials/exploratory-data-analysis-python

https://www.digitalocean.com/community/tutorials/how-to-convert-data-types-in-python-3-pt

https://seaborn.pydata.org/generated/seaborn.pairplot.html

https://www.statology.org/numpy-count/

https://www.dublincity.ie/dublin-city-development-plan-2016-2022/16-development-standards/1638-car-parking-standards/16389-design-criteria

https://scikit-learn.org/stable/

https://scikit-learn.org/stable/tutorial/basic/tutorial.html

https://www.sharpsightlabs.com/blog/scikit-train\_test\_split/

https://www.datacamp.com/tutorial/decision-tree-classification-python

https://www.techtarget.com/searchenterpriseai/definition/data-splitting#:~:text=In%20a%20basic%20two%2Dpart,after%20the%20training%20is%20done.

<https://www.ibm.com/docs/en/spss-modeler/saas?topic=dm-crisp-help-overview>

https://www.investopedia.com/terms/b/binomialdistribution.asp

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.core.groupby.DataFrameGroupBy.describe.html

https://nycadventurebootcamp.com/pmuibhf/pyspark-example-dataframe

https://docs.python.org/3/library/functions.html?highlight=sum#sum

https://docs.python.org/3/library/math.html

https://docs.python.org/3/library/locale.html?highlight=loc#module-locale

https://docs.python.org/3/library/numbers.html

https://docs.python.org/3/tutorial/errors.html

https://www.datacamp.com/tutorial/decision-tree-classification-python

https://thecleverprogrammer.com/2020/12/06/machine-learning-projects-on-time-series-forecasting/