MACHINE LEARNING FOR DATA ANALYSIS

Written Report

By

Muhammad Muneeb Khan

Table of Contents

1. Introduction

What is CRISP-DM Methodology

The CRISP-DM technique, also called the Cross-Industry Standard Process for Data Mining, is a statistics evaluation framework advanced within the Nineties by researchers and practitioners from both academia and industry.

This methodology gives a comprehensive and standardized approach for fixing commercial enterprise issues the use of information analysis. The CRISP-DM framework consists of six stages that are:

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling
- Evaluation
- Deployment

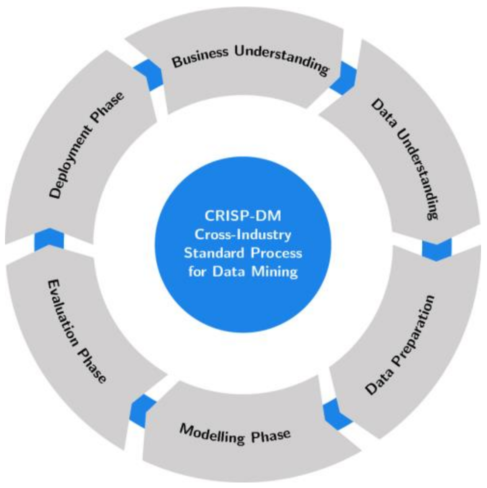


Figure 1 CRISP-DM cross-industry standard process for data mining (Giang Nguyen,2019)

1- **Business Understanding**: This phase includes defining the commercial enterprise hassle and objective, figuring out the dreams and success criteria of the project, and developing an undertaking plan. the principal goal of this segment is to apprehend the business necessities and make sure that the venture aligns with the overall commercial enterprise method.

2- **Data Understanding**: This section involves exploring and studying the available statistics resources to apprehend their excellent, shape, and content material. The purpose of this segment is to decide the suitability of the facts for the challenge and become aware of any data troubles that need to be addressed.
3- **Data Preparation**: This segment entails cleansing, reworking, and integrating the statistics to make it suitable for analysis. the main goals of this segment are to get rid of mistakes, outliers, and beside the point records, and to format the information right into an appropriate structure for evaluation.
4- **Modelling**: This segment entails making use of diverse facts analysis techniques to become aware of styles, relationships, and insights in the statistics. the main goal of this segment is to generate hypotheses and test them using statistical techniques and algorithms.
5- **Evaluation**: This phase includes evaluating the results of the analysis to decide their validity and usability. the main goal of this section is to decide if the outcomes can be used to make choices and clear up the business hassle.
6- **Deployment**: This section entails communicating the consequences of the evaluation to stakeholders and integrating the results into commercial enterprise methods. the principal objective of this phase is to make certain that the effects of the evaluation are used to make knowledgeable business selections.

The CRISP-DM methodology is widely utilized in various industries, along with healthcare, finance, retail, and advertising and marketing, to remedy business issues using information evaluation. Its importance lies in its comprehensive and standardized method, which gives a roadmap for statistics evaluation projects and ensures that projects are finished efficaciously and effectively. further, the CRISP-DM methodology guarantees that everyone applicable aspects of a undertaking are taken into consideration, from defining the commercial enterprise hassle to deploying the consequences, making it a precious tool for facts analysts.

In conclusion, the CRISP-DM method is a complete and standardized approach for fixing business problems the usage of data analysis. Its software and significance are widely diagnosed in diverse industries, and its use has been shown to increase the performance and effectiveness of statistics analysis tasks.

2. Data Understanding, Data Pre-processing, Exploratory Data Analysis

A survey statistics set is a collection of statistics that is collected via a survey. A survey is a tool used to accumulate information from a set of people. Surveys are frequently used to gather records on attitudes, behaviours, and reviews. The survey facts set will generally include a massive wide variety of observations and variables that describe the survey individuals and the responses they provided to the survey questions.

The shape of the survey records set will rely upon the type of survey and the records collection approach used. as an example, a paper-based totally survey will generally have a exceptional structure than a web survey. The survey records set can also comprise demographic information approximately the members, along with their age, gender, and training level. it can additionally incorporate statistics at the responses to the survey questions, consisting of multiple-choice responses, open-ended responses, or rating scales.

In phrases of first-rate, the survey information set might also comprise missing values or inconsistent values. as an instance, a participant may not have answered a question or can also have provided an inconsistent response. The analyst will want to evaluate the quality of the facts and determine if any cleansing or correction is important. The analyst might also need to verify the validity of the data with the aid of move-checking the statistics with outside resources.

In terms of content, the survey records set will include statistics at the variables of hobby, inclusive of attitudes, behaviours, and critiques. The analyst will want to study the data in element to benefit a deep information of the content and to decide if any pre-processing is necessary. for example, the analyst can also need to convert categorical variables into numerical variables or rescale variables to make certain they have the equal scale.

In end, the survey statistics set is a treasured source of facts that may provide insights into the attitudes, behaviours, and critiques of a collection of human beings. The analyst will want to perform facts information, statistics pre-processing, and exploratory records evaluation to make the facts suitable for evaluation and to advantage precious insights from the facts.

In this phase, existing data are explored and explored to understand their quality, structure and content. The purpose of this step is to ensure that the data is adequate to solve a specific problem and to identify data issues that require attention. This includes checking for missing values, understanding the distribution of data, and validating data accuracy. Pre-processing of

data is his third step in the CRISP-DM process. This step cleans, transforms, and integrates the data to make it suitable for analysis.

This may include removing errors, outliers, and outliers, and formatting the data into the correct structure for evaluation. The main purpose of this step is to prepare the data for further analysis and ensure that it is in a usable format.

Exploratory Data Analysis (EDA) is a type of data analysis to better understand the data structure and relationships between variables. This is an important part of the CRISP-DM process. It allows you to discover patterns, relationships, and insights in your data. During the EDA phase, various data analysis techniques such as visualization and statistical techniques are used to test hypotheses and gain a deeper understanding of the data.

This step is also important for identifying potential outliers and data issues that can affect the analysis results.

Attributes For The Survey And The Presentation Of Table

#	Attribute Name	Numerical (N) or Categorical data (C)	Nominal Data (N) or Ordinal Data (O)	Data Type
1	What is your biggest reason for learning to code?	C	N	Str
2	What methods have you used to learn about coding? Please select all that apply.	C	N	Str
3	Which online learning resources have you found helpful? Please select all that apply.	C	N	Str

4	If you have attended in-person coding-related events before, which ones have you found helpful? Please select all that apply.	C	N	Str
5	If you have listened to coding-related podcasts before, which ones have you found helpful? Please select all that apply.	C	N	Str
6	If you have watched coding-related YouTube videos before, which channels have you found helpful? Please select all that apply.	C	N	Str
7	About how many hours do you spend learning each week?	N	N	Int
8	About how many months have you been programming?	N	N	Str
9	Aside from university tuition, about how much money have you spent on learning to code so far (in US Dollars)?	N	N	Int
10	Are you already employed in a software development job?	C	N	Str
11	If you are already employed as a developer, is this your first software development job?	C	N	Bool
12	If you are NOT already a developer, are you interested in a software development career?	C	N	Bool
13	If you are interested in a software development career, would you prefer to...	C	N	Str
14	Which of these careers are you interested in?	C	O	Str
15	When do you plan to start applying for developer jobs?	C	N	Str
16	About how much money do you expect to earn per year at your first developer job (in US Dollars)?	N	O	Float
17	Please select up to 3 reasons why you are interested in a software development career	C	O	Str
18	After the pandemic, how many days would you ideally like to work from home versus in an office each week?	N	N	Int
19	Are you willing to relocate for a job?	C	N	Ternary
20	Regarding employment status, are you currently working	C	N	Bool
21	If you are currently working, which field do you work in	C	O	Str
22	About how much money did you earn last year from any job or employment (in US Dollars)	N	N	Float

23	How old are you?	N	N	Float
24	Which of the following best represents how you think of yourself?	C	N	Str
25	With which of these groups do you primarily identify?	C	O	Str
26	Which part of the world do you live in?	C	N	Str
27	If you are living in the US, which state do you currently live in?	C	O	Str
28	About how many people live in your city?	N	N	Int
29	Is your country of citizenship different from the country where you live?	C	N	Bool
30	Are you an ethnic minority in your country?	C	N	Bool
31	Is English your second language?	C	N	Bool
32	What is the highest degree or level of school you have completed?	C	O	Str
33	If you attended a university, what did you study?	C	O	Str
34	Are you currently a student attending or enrolled in regular school, that is in an elementary school, a middle school, a high school, a college, or a graduate school?	C	O	Str
35	What's your marital status?	C	N	Bool
36	How many children do you have? By children, we mean any biological, step, or adopted children.	N	N	Str
37	Do you financially support any dependents, older relatives or relatives with disabilities?	C	N	Str
38	Do you think you have enough savings to survive for 3 months with no income?	C	N	Bool
39	How much debt does your household have? [Car loans]	N	O	Float
40	How long have you been working in your current job?	C	N	Str
41	Before you got your last job, how many months did you spend looking for a job?	N	N	Float
42	If you are working, thinking about the next 12 months, how likely do you think it is that you will lose your job or be laid off?	C	N	Bool
43	If you are working, how easy would it be for you to find a job with another employer with	C	N	Str

	approximately the same income and fringe benefits you now have?			
44	Do you consider yourself under-employed? (Under-employment means working a job that is below your education level.)	C	N	Bool
45	Please tell us how satisfied you are with each of these following aspects of your present job [Earnings]	C	O	Str
46	Have you served in your country's military before?	C	N	Bool
47	Do you currently receive disability benefits from your government?	C	N	Bool
48	Do you have high-speed internet at your home?'	C	N	Bool

- **We will use the following and is identified as important data:**

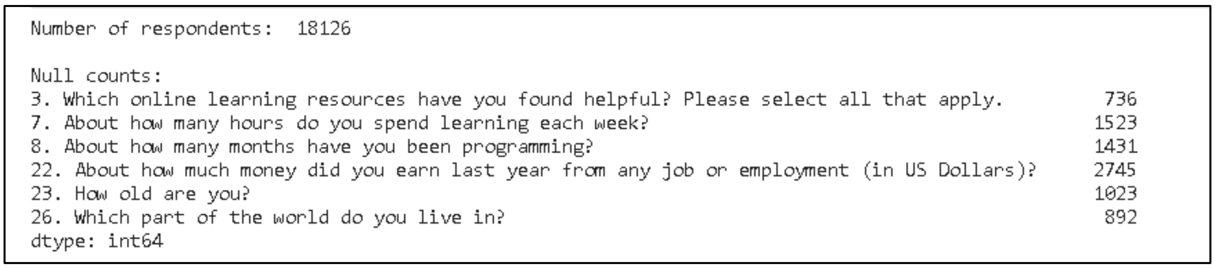
#	Description	Attribute name	Data Type
1	Online resources used	Which online learning resources have you foundhelpful?	**Object**
2	Region	The Region that the developer lives in.	**Object**
3	Age	The developers age (in years)	**Float**
4	YearsCodePro	The number of months been coding (Converted to the years)	**Float**
5	WorkWeekHrs	The number of hours the programmer studies per week.	**Float**
6	Income	The salary of the developer converted to USD.	**Float**
7	Education Level	How much of the Education level achieved	**Object**
8	Interest	What the developer is most interested in	**Object**

• Cleaning The Data

When working with raw data, it's common to encounter missing values and other issues. These missing values can create problems later on because they can impact the accuracy of the models. As mentioned in the introduction, bad analytics can lead to bad business decisions, highlighting the importance of having clean and relevant data. Our complete dataset contains 18,126 rows and 63 columns, resulting in 1,141,938 data points. With a dataset of this size, it's expected to have missing data, and we have identified 156,066 such values that do not contribute to our research goal. These missing values are not helpful and can even be harmful to our analysis.

To ensure that our data is accurate, data cleaning is necessary before conducting any data analysis. We need to remove any unnecessary data that may lead to false or inaccurate results.

The first step in the data cleaning process is to create a new data frame that contains only the essential attributes that we need for our analysis. The dataframe we're investigating total of rows of 18,126 and on count there are 7 columns, but when we add all the entries we're leftover with the sum of 126,882 data points. This means that we have eliminated 1,015,056 data points which might be null or hypothetical or irrelevant of what we're doing. Using these datapoints when building the models, it would negatively impact the accuracy of our results. Therefore, it's crucial to have clean and relevant data for successful data analysis.



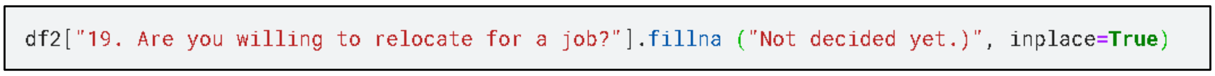
```
Number of respondents:  18126

Null counts:
3. Which online learning resources have you found helpful? Please select all that apply.        736
7. About how many hours do you spend learning each week?                                        1523
8. About how many months have you been programming?                                             1431
22. About how much money did you earn last year from any job or employment (in US Dollars)?     2745
23. How old are you?                                                                            1023
26. Which part of the world do you live in?                                                      892
dtype: int64
```

Figure 2 – Result of executing the figure 2.

Another column in our dataset, "Are you willing to relocate," has a total of 760 missing values. This means that we don't have information about whether these respondents are open to moving to a new location for work. Since this missing data can affect the accuracy of our models, we need to find a way to handle it.

One approach is to impute the missing data with the most common response from other participants. However, in this case, imputing the data wouldn't make sense because we can't assume that most respondents are willing to relocate. It's important to note that any imputation method can introduce bias into our data, so we need to be careful when deciding how to handle missing values.

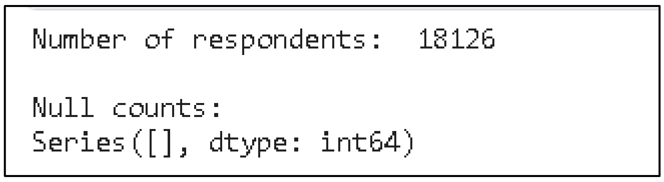
Instead of imputing, we can assume about the missing values. In this case, we can assume that those who didn't respond to the question are not willing to relocate. This assumption might not be accurate for every respondent, but it's a reasonable starting point. Before making this assumption, we had 11,366 respondents who answered the question, and after if the 760 missing values indicate a lack of willingness to relocate, we end up with a total of 12,126 respondents who are not willing to relocate.



```
df2["19. Are you willing to relocate for a job?"].fillna ("Not decided yet.)", inplace=True)
```

Figure 3 - Filling the null rows to make sure we've appropriate values.

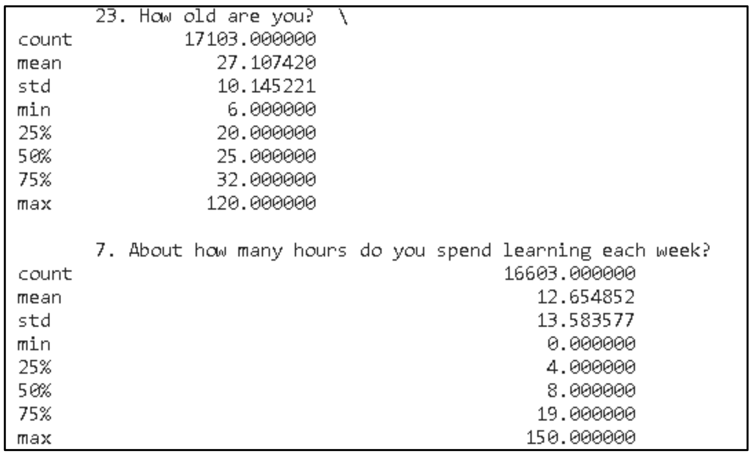
In figure 3 we've used fillna() function to fill the appropriate values.



```
Number of respondents:  18126

Null counts:
Series([], dtype: int64)
```

Figure 4 - After we've removed all the null values.

In the above we can see that there's no null data left, so we can move on to the next step now which is to find the numerical attributes of our data,



```
       23. How old are you?  \
count          17103.000000
mean              27.107420
std               10.145221
min                6.000000
25%               20.000000
50%               25.000000
75%               32.000000
max              120.000000

       7. About how many hours do you spend learning each week?
count                                       16603.000000
mean                                           12.654852
std                                            13.583577
min                                             0.000000
25%                                             4.000000
50%                                             8.000000
75%                                            19.000000
max                                           150.000000
```

Figure 5 - Displaying the range of the numerical attributes.

In figure 5 we can clearly see the numerical attributes of the age and the Hours spend learning which includes the total values, mean value, min, max and other mentioned in the figure.

The purpose of describing the numerical attributes is to detect any unusual values or outliers that do not follow simple logic and are implausible. One example is the maximum value of "hours spent learning each week" which is 150 hours. This is an unrealistic value considering that a week only has 150 hours. This seems very impossible for an induvial to spend this much of their time in a week solely on how to learn programming or to code. Another instance of outliers is found in the column of "How old are you?" where minimum, maximum points are present. On counting minimum value of '6' is improbable since it is a very young for someone to start and learn the coding and programming related stuff. So it is safe to assume this would be an outlier. The maximum value of '120' is also unrealistic since the oldest person in the world is only 115 years old.

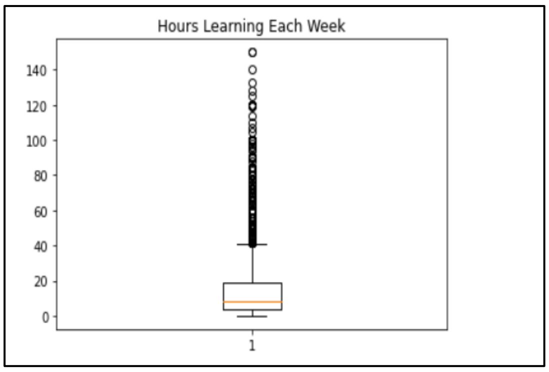


Figure 6 - Boxplot for hour learning each week.

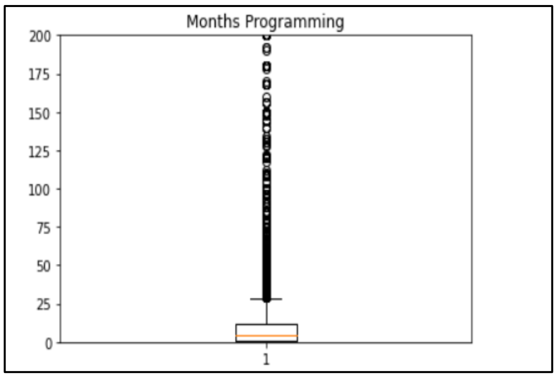
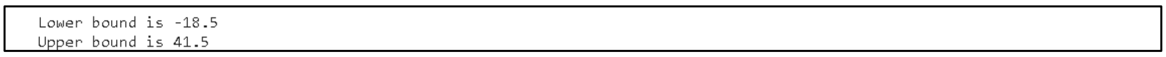


Figure 7 - Boxplot for month programming.

On counting we know that , maximum hours in a week are 168, and when we count the average of it, people spend around 56 hours sleeping, 9 hours eating, and 3 hours commuting. Therefore, the average person has around 100 hours of free time per week. Any value above this number should be considered an outlier and dealt with before building models.

When examining the age attribute using a box plot, we noticed several irregularities. On a quick google search we got to know that the oldest person to live in this decade was just the age of 115, so it would be safe to assume that any value that exceeds this should be marked as outlier. Similarly, now it’s same to assume that the bound should be set to 95, as it is highly unlikely that anyone could live longer than that. For the lower bound, it was observed that the minimum age recorded was 6 years old, which is highly unlikely for someone to be learning how to code at that age. Therefore, anything below the age of 10 should be considered an outlier and treated accordingly.



```
Lower bound is -18.5
Upper bound is 41.5
```

Figure 8 - Displaying the current range and bounds for hour learning.

As you can see in the image, the Turkish rules can be a bit hypothetical and inappropriate for this case. A lower bound on the "how big" the dataset is unlikely to be possible because at the age of 2 you just can't use a computer effectively, but an upper bound makes sense because usually people learning to code simply don't. On further research we got to know that an average age for a person to retire in the UK is of 65, and in fact, so this seems to be very unrealistic that someone above that age wants to stay in front of a computer and learn to code. So, it is safe to assume that the age bound can be set from 10 to 75 years of age.

Next, we can Since you can't actually spend time doing nothing, the lower limitation for "hours spent" is also illogical.

Because people frequently have obligations during the week, such as attending college or working, a cap makes much more sense. While the amount of hours spent genuinely relies on where the responder resides, the variety of hours does not alter. Respondents might devote 100 hours per week coding if they had no other obligations, which is unusual but not impossible. As a result, the "Hours each week" range stays constant from 0 to 100 hours.

- **Removing the unnecessary and null data from our table**

We will not set a realistic bound on our data and remove any of the null or unnecessary data from our data using drop.

• Exploratory data

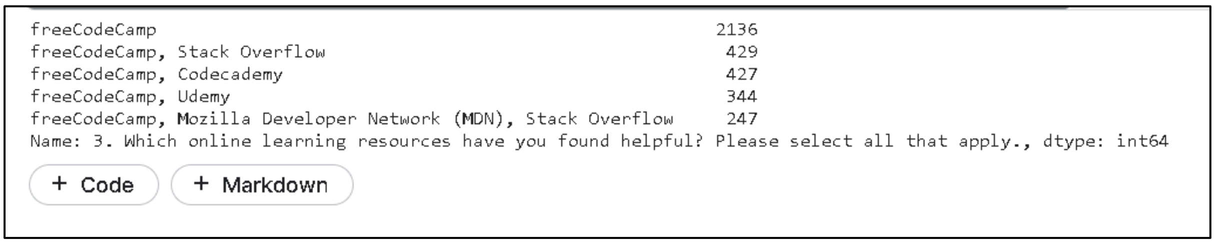
Exploratory data analysis is a method of analysing data by conducting investigations into it to discover patterns or spot outliers. The following section explores some important characteristics for understanding the data.

Education Level (EdLevel)

To start analysing education level, we need to look at the data stored in the attribute. The feature includes nine different options from which developers can choose one. We can visualize the data with a horizontal bar graph.

This led to the option baccalaureate being chosen the most frequently, whereas option primary/primary was chosen the least frequently. We may infer that the majority of Stack Overflow programmers are of a certain type.

We got the following output for the 3rd column to see if it has any values:



```
freeCodeCamp                                                  2136
freeCodeCamp, Stack Overflow                                   429
freeCodeCamp, Codecademy                                       427
freeCodeCamp, Udemy                                            344
freeCodeCamp, Mozilla Developer Network (MDN), Stack Overflow     247
Name: 3. Which online learning resources have you found helpful? Please select all that apply., dtype: int64
```

Figure 9 - Most frequent values in the column.

Plot graph of the above we got the following:

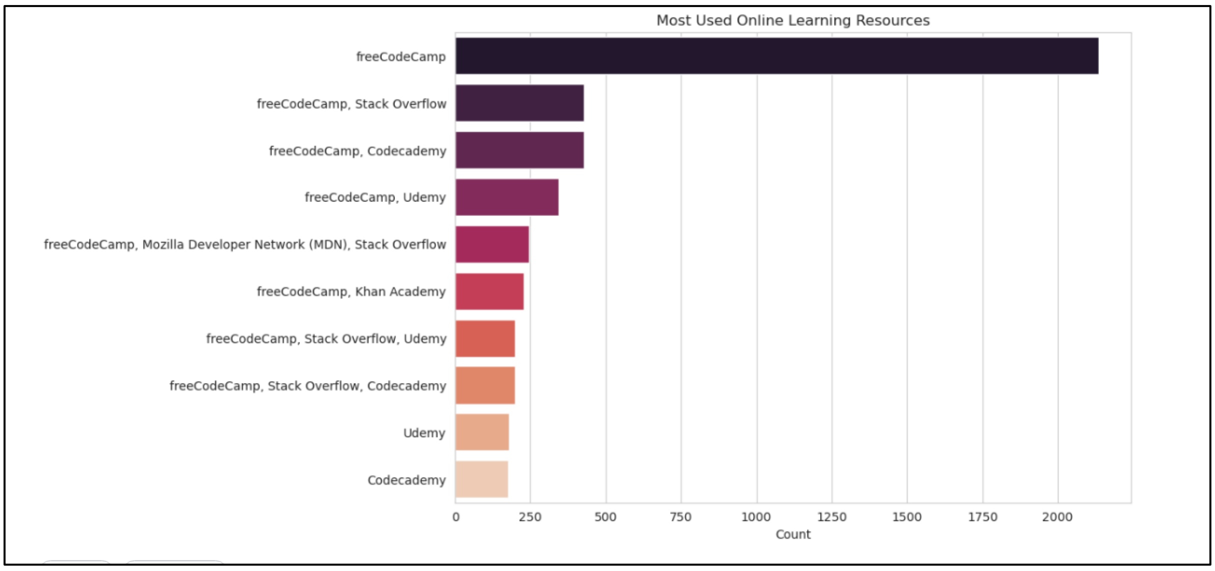


Figure 10 – Horizontal bar chart to show the most used resources.

The most frequent value, as seen on the horizontal bar graph above, is freeCodeCamp. FreeCodeCamp, however, is mentioned in all other values, showing that it is a website that respondents commonly visit. The least chosen option was the Udemy Network, as freeCodeCamp and Stack Overflow were the third most chosen options. This shows that udemy uses the least. For better understanding of the above chart please see the figure 17 below.

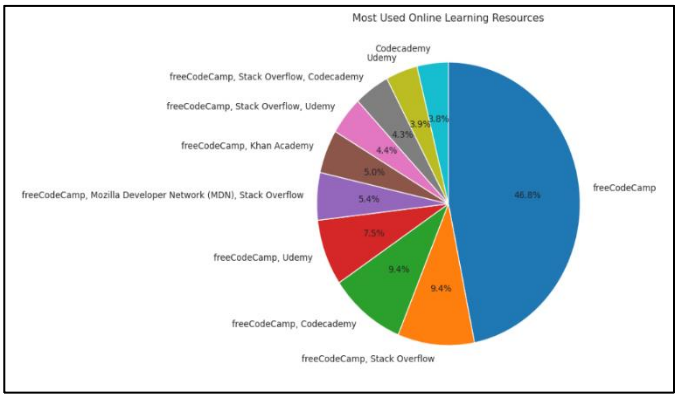


Figure 11 - Pie chart to display the most used resources.

To see the percentages, it's easier to see in a pie chart. From there, we can see the developers used the freeCodeCamp most which made up to the 52.5% of the total that is more than rest of the values whereas the least was Udemy which remained on 8.4% whereas we've to keep in mind developers still used the freeCodeCamp in compliance with the other resources so we would have to count that in too.

Country

There are 183 variables in the country attribute, and each of them has at least one choice made by a developer. The following will let us see how many developers there are in each nation:

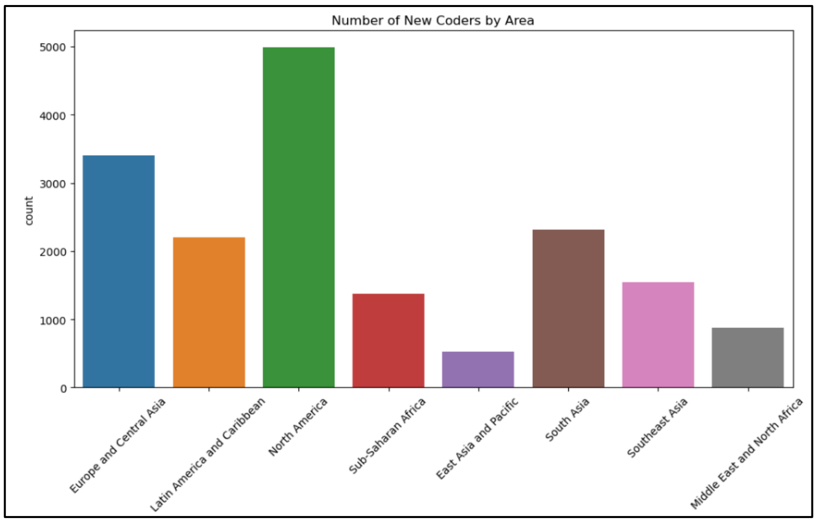


Figure 12- Graph for Programmers in each Region

To extend our analysis, we can compute the proportion of developers from each country using a pie chart. To enhance the visual display of the chart, we can group all countries with a proportion below 2.4%. A careful inspection of the resulting pie chart reveals that the United States has the largest number of Stack Overflow developers, with a significant margin.

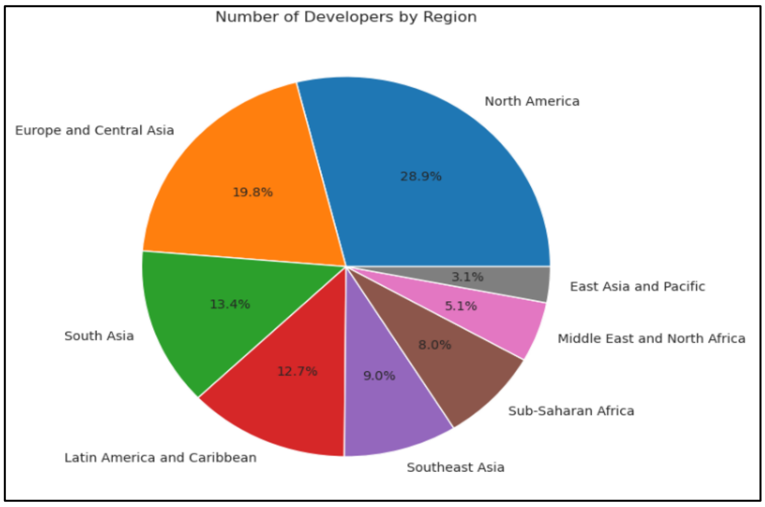


Figure 13 – Pie chart to display the number of developers from each region.

Age

The survey question on Stack Overflow that captures the age of developers accepts any numerical response, which can be stored as floating-point numbers. But it is dependent on the developers to provide realistic values, and it is possible to get incorrect or unrealistic data in the response. This has resulted in a smaller dataset with only 45,446 respondents to this question. Before delving into the analysis, it is important to identify any outliers in the dataset and remove them if necessary. One approach to achieve this is to create a graph to visually spot outliers and anything that seems not very possible to be achieved by an ordinary human being.

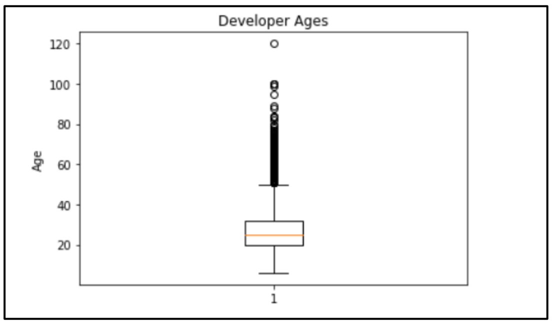


Figure 14 - Age of developers.

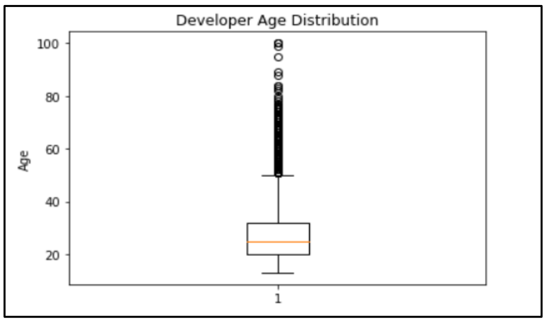


Figure 15 – Age of developers with restrictions

Here we’ve set a limit to the minimum and the maximum age because logically it’s impossible for someone to be a developer before the age of 13 and more than the age of 100 years because there was only 1 value that was breaching this age limit and the age was 250 years old which seems not possible. These are counted as outliers whereas rest of the values are very logical and possible for a human.

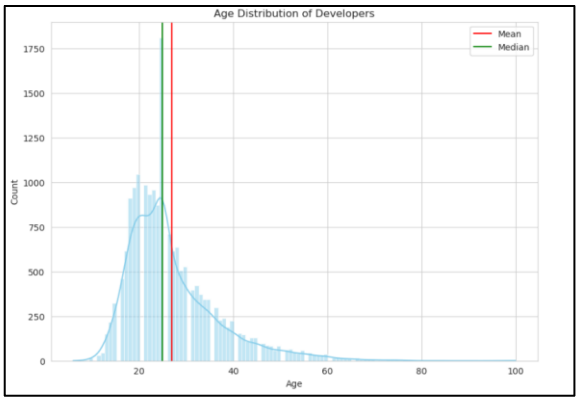


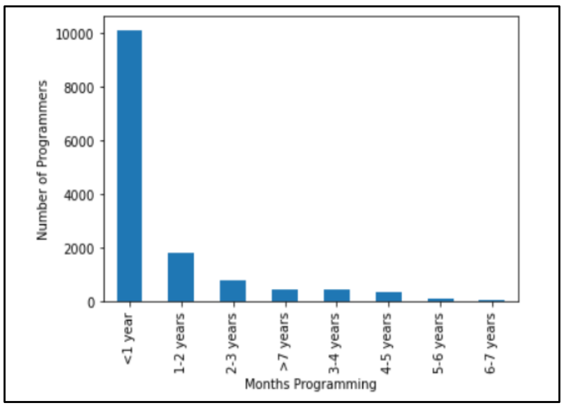
Figure 16 - Histogram to display the age of the developers.

Here we’ve used a histogram to display the age of the developers and we’ve also defined the mean and median age to make our histogram look more presentable and easier to understand.

This shows that most of the programmers are between the age of 20-30 whereas the least number of developers can be seen in the age of 80.

Months programming

When we see the data within the “About how many months have you been programming?” we can see that there are a lot of variations. Other than that, it would be really hard if we were to visualize the data for each month and there are a lot of data to be presented for this, we will have to convert the months to the years and then present which would be a lot easier to represent on the graph.



Here we can see that most of the programmers have started nearly a year ago whereas there’s very few of them who have the experience of more than 6-7 years in fact they’re the lowest in the histogram.

Figure 17 - Histogram showing years programming and number of programmers.

This also proves that many people have recently got in to the field of programming this is due to extreme popularity of computers and technology recently.

In the above we can only see a rough sketch of the years programming, now let's draw a pie graph and get a detailed view of the data.

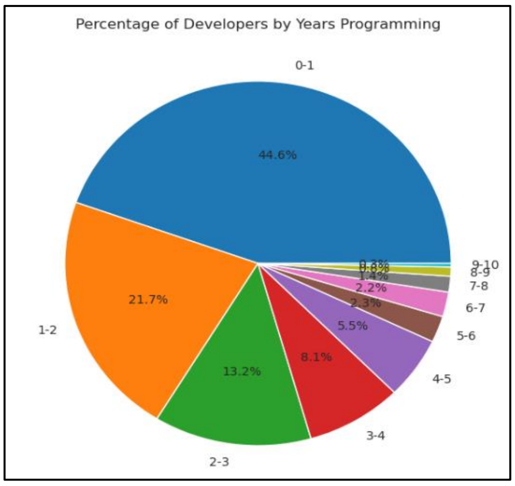
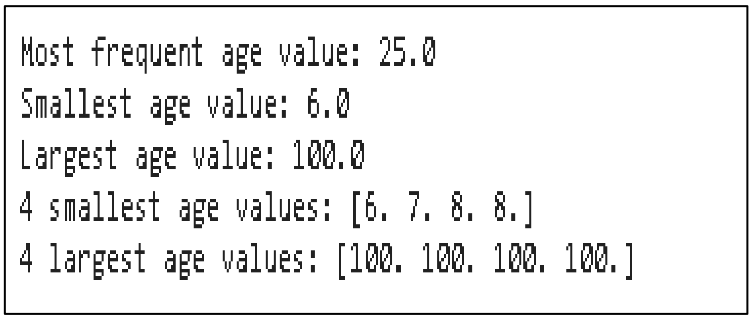


Figure 18 - Pie chart displaying percentages of programmers and months coding.

This pie chart shows the percentage with a conventional signs chart to make it look more presentable and easier to understand due to large amount of data in our file.

Using this pie chart, it's safe to assume that more than 80% of programmers started programming in recent 1-2 years and they're not someone we can say very expert, but this is very good for the industry as there are a lot of new people joining.

Now we will generate the Most frequent values, smallest values and the largest values.



```
Most frequent age value: 25.0
Smallest age value: 6.0
Largest age value: 100.0
4 smallest age values: [6. 7. 8. 8.]
4 largest age values: [100. 100. 100. 100.]
```

Figure 19 - Frequent, Smallest, and largest values.

Hours worked per weeks:

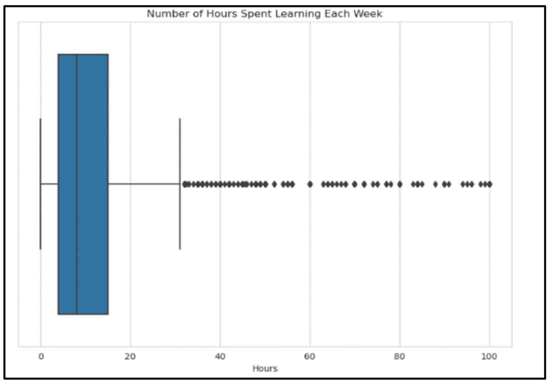


Figure 20 - Boxplot of hours worked learning/week.

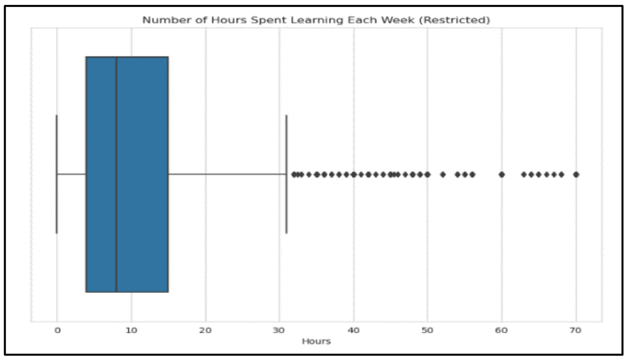


Figure 21 – Restricted Boxplot for the figure 26.

In the above figure we can clearly see that there's a visible number of developers that have put number of hours worked to study more than 100 which seems very impossible to achieve because there's only 168 hours in a week in which we've to sleep and eat too. So, what we did in figure 27 was that we put a realistic limit of 100 hours

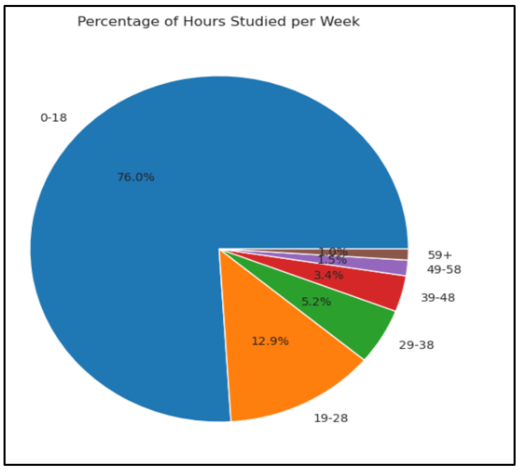


Figure 22- Pie Chart of hours studied per week.

Using this pie chart, we can easily categorize the data and see the percentage of hours that have been studied per week. More than 70% of developers have studied around 0-19 hours a week whereas 1.6% a very minor amount of devs. studied 60-100 hours a week. From this we can guess that average amount spent on studying coding each week is around 20-40 hours a week.

A scatter matrix is a powerful tool that allows us to visualize the relationships between multiple variables in a single plot. In the code above, we have used the scatter matrix to plot the relationships between the variables of Months of programming, Age, and Studying per week. The scatter matrix helps us to identify any trends or patterns in the data, which may help us to draw insights from the data.

However, before creating the scatter matrix, we needed to ensure that the data was clean and free from any hypothetical values. In the code above, we removed any null values from the data and set limits on certain variables, such as hours worked per week and years of coding professionally.

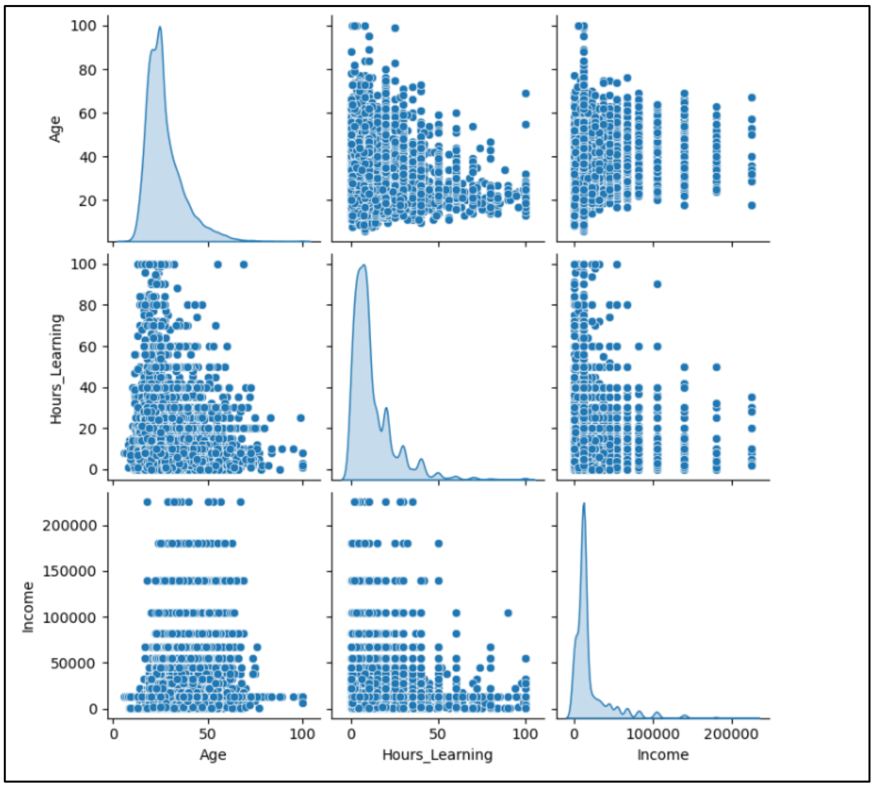


Figure 23 - Scatter Matrix between Age, Hours per week and Income.

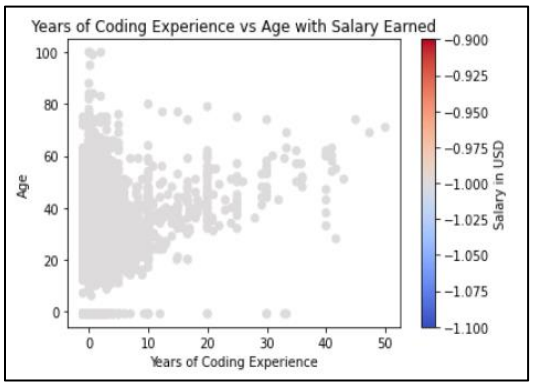


Figure 24 - Scatter plot demonstrating years coding and the salaries earned.

In the figure 30 we've used the scatter plot to show the relation between the years coding and the age with the salary earned. This shows that the more older you're, the more you've the experience there's likely the higher chance of you getting the more salary. The only problem with this graph is that even after cleaning there could still be the outliers because of different developers from different countries resulting in very odd average salaries but this is minority, so the rest of the plot is self-explanatory.

In the figure we drew a correlation heatmap to visualizations to explore relationships in a dataset (df) using the Seaborn and Matplotlib libraries and it is created with a 10x7 size, displaying the pairwise correlations between variables. The heatmap uses a cool-warm color map, with annotations showing the correlation values rounded to two decimal places and 0.5-line width separating the cells. A title "Correlation Heatmap" is added and the plot is displayed.

A pair plot is generated, showing scatter plots for each pair of variables in the dataset, with kernel density estimates on the diagonal. The plot is displayed.

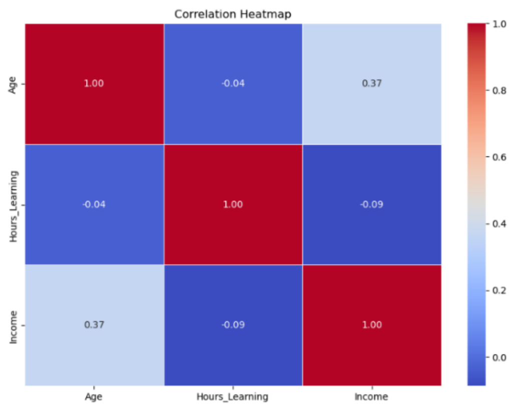


Figure 25 – Correlation Heatmap

Section 3 Cluster Analysis

Finding clusters within a dataset that can be further examined is possible with the help of cluster analysis. In this instance, we're going to split our information into two groups depending on if the developer's earnings are higher or lower than the typical wage of $54,289 in order to analyze data. As these are the goal factors and have the potential to disturb the clusters, we are removing the revenue and salary-made columns from our data frame.

Hierarchical clustering is less effective and yields less accurate findings, but we'll just use k-means clustering instead. Several factors of the k-means clustering technique can influence the analysis's findings:

- n clusters = 3: Users chose this value because it generated findings that were simpler to interpret than other numbers. In order to corroborate this, we additionally utilized the elbow technique, which revealed that the elbow is already at 3 clusters.
- Init = "random": This option determines the initialization strategy and is set to "random" because, despite being less effective, it arbitrarily chooses n clusters rows.
- Init = "random": This option determines the initialization strategy and is set to "random" because, despite being less effective, it arbitrarily chooses n clusters rows.
- With n init set to 10, the technique is executed ten times with various center seeds, with the optimal outcome coming from all of the runs. It's usually set to 10.max iter = 300.
- tol = 1e-4: This value, which represents the variation in the cluster centres between rounds, is usually set to 1e-4.

- To maintain uniformity across all groups, the parameter random state is assigned to 13, an arbitrary integer, when initialising the centroid. algorithm = "auto, traditional EM algorithm, that may be adjusted to "full," is an option.

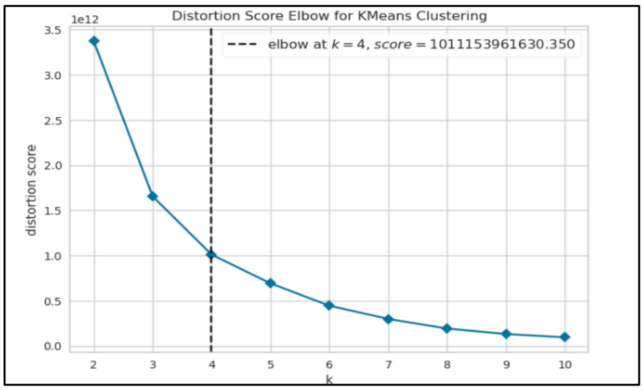


Figure 26 - Low-income elbow visualiser for cluster analysis.

Here it will plot the elbow curve, with the number of clusters on the x-axis and the inertia (within-cluster sum of squares) on the y-axis and annotates the plot with the elbow point. The goal of this analysis is to identify patterns and groupings in the data, specifically among respondents who reported low income. Then it will pre-process it and performs KMeans clustering on a subset of the data to identify an appropriate number of clusters.

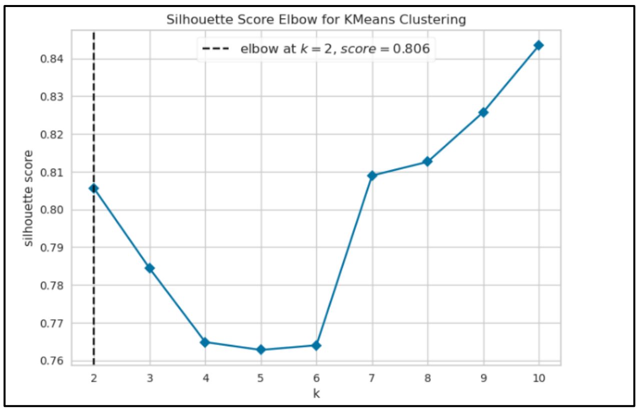
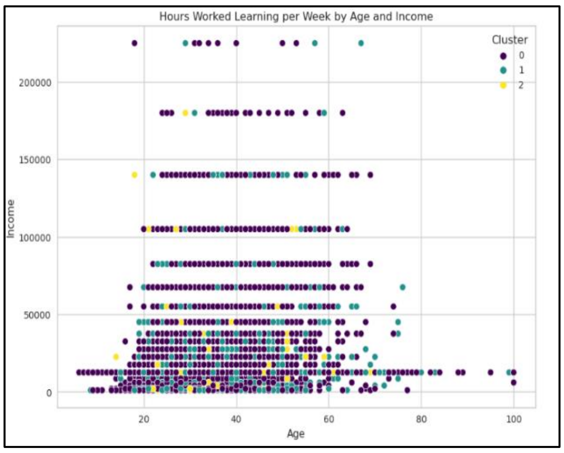


Figure 27 - High-income elbow visualiser for cluster analysis

For high-income cluster analysis, we will follow a similar approach as for low-income. First, we will plot an elbow curve with the number of clusters on the x-axis and the inertia (within-cluster sum of squares) on the y-axis. The goal is to identify patterns and groupings in the data, specifically among respondents who reported high income. The curve will help us determine the appropriate number of clusters to use in KMeans clustering. We will preprocess the data and then apply KMeans clustering to identify the clusters.

Earning of the developers Analysis

KMeans clustering with k=3 on the 'WorkWeekHrs' and 'YearsCodingProf' columns of the 'low_income_df' dataframe. We then add the cluster labels to the dataframe and plot a scatter plot using the seaborn library, with the 'WorkWeekHrs' column on the x-axis, the 'YearsCodingProf' column on the y-axis, and the cluster labels determining the color of the points.



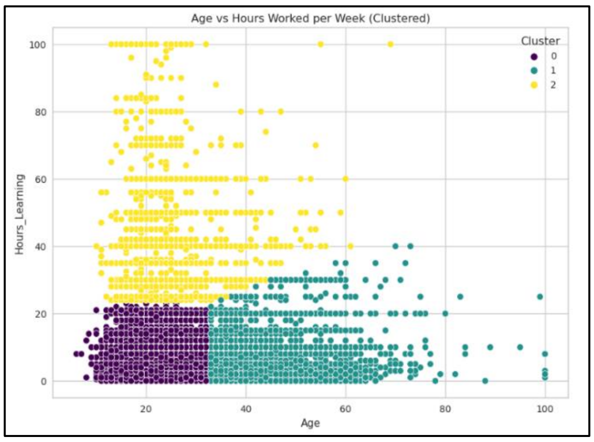


Figure 28 – Scatter plot with clusters for hrs/week

Figure 29 – Scatter plot with clusters for age/years spent.

When we look at the figure 32, we can see the clusters clearly. 3 different clusters can easily be seen on the figure which are used to identify the difference easily. This tells us that those developers who put more hours into the studies are most likely to be the highly experienced ones, whereas those don't put much effort in learning are less qualified and the last category are of those less experienced developers who work a lot, these are soon going to land in the first cluster respectively.

Whereas in the figure 33 we used the age data and divided into the 3 clusters too, here these

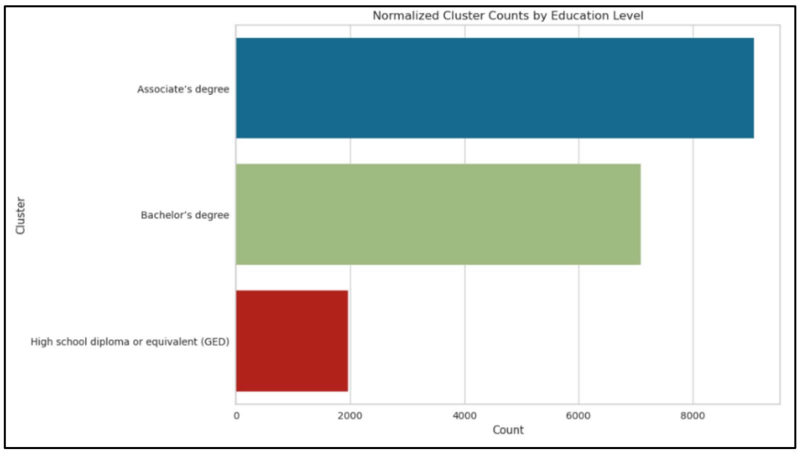


Figure 30 – Cluster count by the Education Level

In the figure 34 we can easily compare the clusters and see that the first one has more values than anyone else whereas there's negligible in the 3rd which according to the index is the 2nd cluster represents the young people who don't work as much of those

represent the old developers that work a lot, young developers that work more, and the cluster 2.

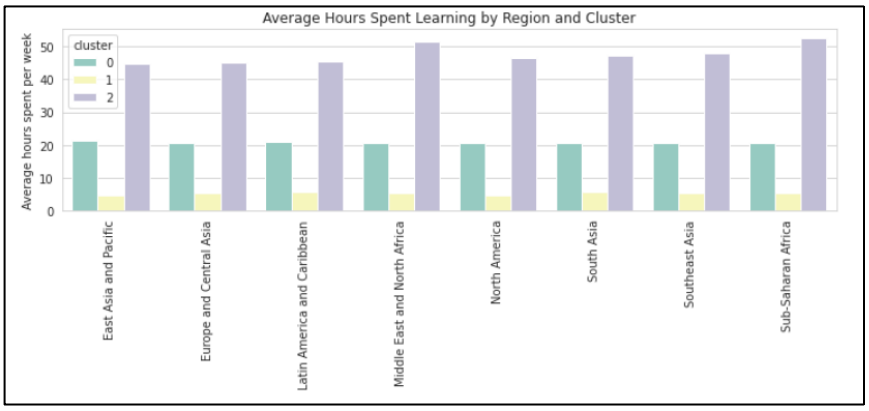


Figure 31 - Normalised bar plot on the regions in which the developerslive in, as per the cluster with the age factor.

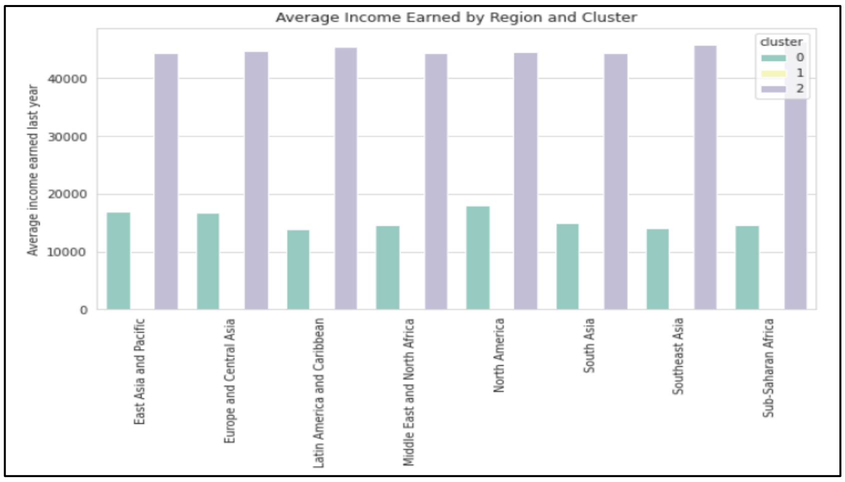


Figure 32- Normalised bar plot on the regions in which the developers live in, as per the cluster with the salary factor.

In figure 35 we perform clustering on the data to identify patterns in the number of hours respondents spend learning per week by region. It uses KMeans clustering algorithm to cluster the data into three groups and produces a bar plot that shows the average number of hours spent learning per week by region and cluster whereas in figure 36 we perform clustering on the data to identify patterns in the income earned by respondents last year by region. It uses KMeans clustering algorithm to cluster the data into three groups and produces a bar plot that shows the average income earned last year by region and cluster. The income data is first converted to a numeric format and any missing values are filled with the value $0.

Summary:

We learned that the low-income group could be split into three categories. Developers in Cluster 0 were the oldest and had the most years of formal coding expertise. They were primarily based outside of Europe and Asia and had the largest proportion of coders with master's degrees. Developers in Cluster 1 were the youngest, worked its most hours per week, and had the least quantity of professional coding expertise. And furthermore, they had the major portion of North American coders. This group made up most coders in Cluster 2 and had the fewest weekly hours worked. Likewise, they were most programmers with a bachelors.

Among the categories, developers in Cluster 0 were the oldest, with the greatest number of years of successful coding expertise and had the largest proportion of master's-educated developers. Developers in Cluster 1 made up the majority of those who didn't reside in the SEA or the NA and had PhD degrees. Most workers in Cluster 2 belonged to this cohort, which was also the youngest. They had the largest proportion of developers from the NA and the least quantity of professional coding experience.

Section 4 – Machine Learning Models

Machine Learning Workflow

Machine learning workflow is for streamlining procedures, enhancing judgement, and gaining insightful data. The issue that needs to be addressed is first identified in the machine learning workflow. Data needs to be gathered and cleaned up from different sources after the issue has been discovered. This entails getting the data ready, altered, and formatted so that it can be incorporated into a programme. Following the completion of this phase, experimental data analysis is carried out to find patterns or trends that can help determine what kind of machine learning model should be used for the best outcomes.

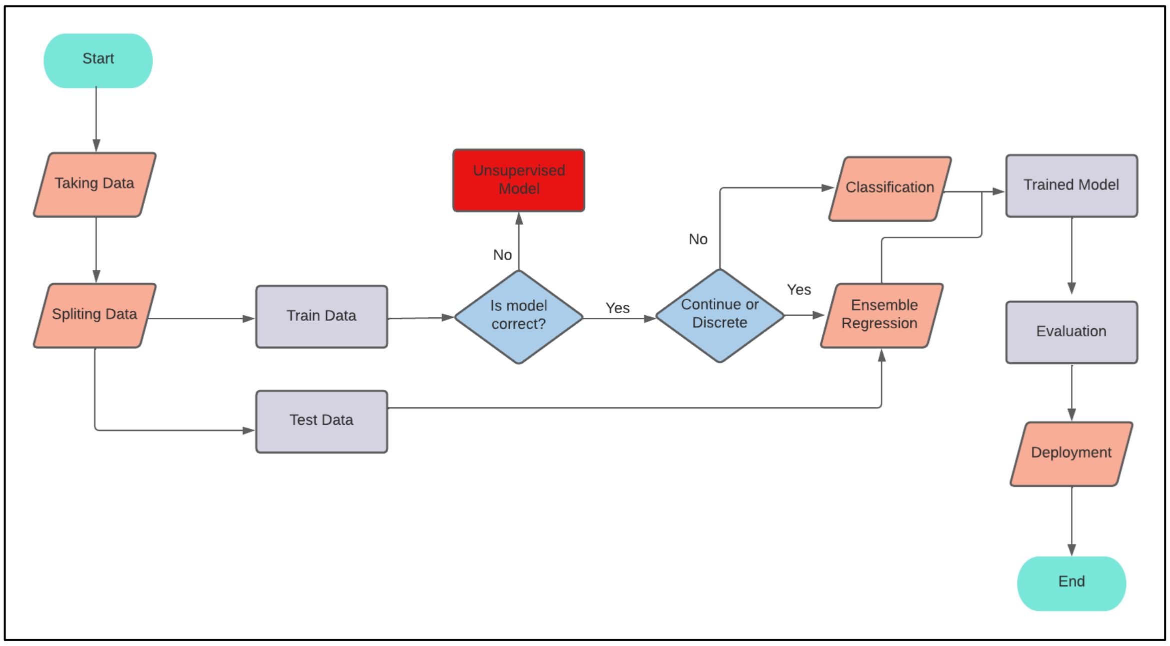


Figure 33 – Flowchart for the machine learning workflow.

Once the model has been identified then it’s time for training the algorithm on labelled datasets which includes fine-tuning parameters as well as feature selection techniques such as selecting important variables from existing ones based on correlations or other metrics like variance inflation factor (VIF).

There are many factors to take into consideration when choosing which algorithm for machine learning to use. The research concern is a segmentation problem because it calls for categorising the response into 2 categories: high income and low income. As a result, classification models are much more ideal for use in classification tasks than regression models. There are numerous factors because one model is preselected over the other, including speed, efficiency, and bandwidth usage. Decision tree classification and linear regression method have been adopted for their quickness and simplicity of interpretation. Whereas the multilayer perceptron (MLP) neural network (ANN) model was chosen for accuracy.

Random Forest Classifier

Random Forest Regression is a widely used machine learning technique that blends numerous decision tree models to establish a potent ensemble learning method. This model constructs a plethora of decision trees by employing bootstrapped training datasets and choosing a random subset of attributes at each node. Subsequently, the model consolidates the forecasts of all the trees to generate the result. As a result of its exceptional precision in managing high-dimensional data, this method is especially useful for handling complex datasets with a wide variety of characteristics. In addition, compared to a single decision tree, the forest-based model exhibits resistance to aberrations and incomplete information and is less prone to overfitting. The forest-based model's interpretability, however, is its main drawback. Being able to understand the prediction process might be challenging because of the mix of many decision trees. In summary, forest-based regression is a valuable instrument for predictive modeling, especially when working with complex and high-dimensional data.

In the given code, multiple classification models are trained, including a Random Forest algorithm. To identify the ideal hyperparameters for this Random Forest model, we carry out a grid search combined with cross-validation. The hyperparameters under examination consist of:

- `n_estimators`: This factor determines the forest's tree count. Our search includes the possibilities of 10, 50, 100, and 200 trees.
- `max_depth`: This factor establishes the tree's maximum depth. We test various depths, such as unrestricted depth (None), 10, 30, and 50.
- `min_samples_split`: This factor sets the lowest number of samples needed to divide an internal node. In our search, we explore the options of 2, 5, and 10 samples.
- `min_samples_leaf`: This factor denotes the least number of samples required at a leaf node. We test different values, like 1, 2, and 4 samples.

By utilizing the `GridSearchCV` function, we evaluate each combination of these hyperparameters and choose the best set for our Random Forest Classifier. It's essential to note that the provided code is for classification purposes, not regression. If you plan to use Random Forest Regression, you should substitute `RandomForestClassifier` with `RandomForestRegressor` and adjust the code accordingly.

Logistic Regression Model

Logistic Regression Model is a technique used to check the occurrence of something in a specified event where there are some independent variables. Outcome in such the model are usually a binary variable and we know that binary values range from 0 to 1 so there's only possibility of two values. Some of the examples of this could be that we can use this model to predict whether a person will buy something based on their age and area.

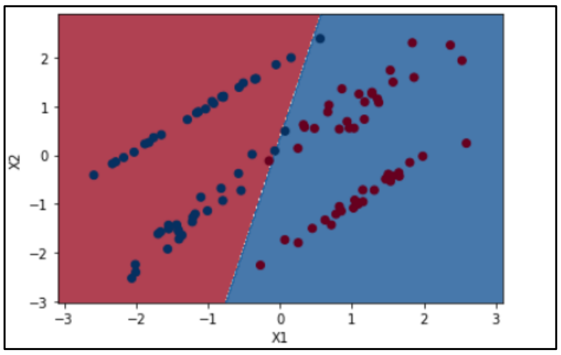


Figure 34 - Figure of the Logistic Regression Model.

We use a mathematical function in this model to call a function which is usually known as Logical function which will map the relationship between how likely that event is going to happen based on the independent variables. As discussed above it will take the value between 0 and 1 and then show the possibility or probability of the thing happening.

In figure 38 we generated the data to with two different features and a binary variable. As we can see in the figure it is divided into two different sections one of which is training and the other one is testing. We test our data in the testing to predict the accuracy of the model. Once we've the accuracy of the model we define the decision boundaries and the spots on our figure which can clearly be seen on the image. Decision boundary is the line that is diving in the centre of our figure. Then we've coloured the data points in dark blue and red. This is calculated by the input variables and the coefficients that our model learns during the training. Then the result can be seen in the classifying data. This model has the accuracy of 0.82%.

Multilayer Perceptron (MLP) Model

MLP is defined as an Artificial Neural Network which has different nodes. In each of these nodes there are different inputs which are processed by a non- linear function which leads to an output. In

MLP the input layer takes the data and then output layer is responsible for producing the output. It also has some hidden layers which are responsible for processing the information. These hidden layers have many computational algorithms working in it which are processing the data received from the input layer. The nodes in the hidden layers can have varying numbers of neurons and can be connected in different ways. Through these nodes we connect these complex layers to produce the result.

We've used this Model in this coursework to predict the classification of different problems or data.

Normalisation is a process used to scale data so that it falls within a specific range, making it easier to compare different variables. While normalisation is not as necessary for neural networks as it is for logistic regression, it has been observed that data that is normalised tends to improve the efficiency of neural networks, leading to better predictions.

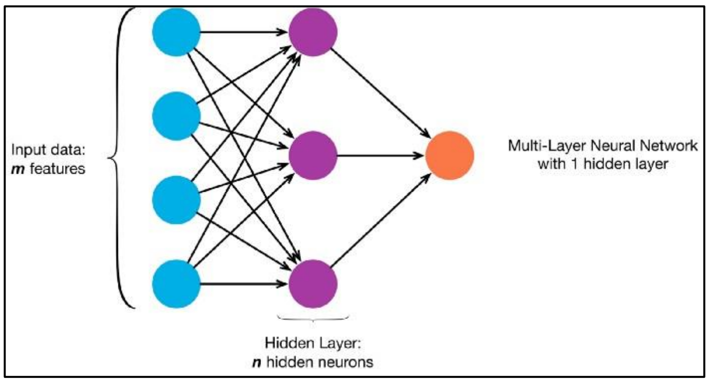


Figure 35 - MLP Neural Network (Machine Learning TV,2017).

In our work, two types of normalization, MinMaxScaler and StandardScaler, were used and tested with the Multilayer Perceptron (MLP) model. The results showed that the StandardScaler normalization method and in the final model we've used this to enhance the accuracy.

As a result of its exceptional precision in managing high-dimensional data, this method is especially useful for handling complex datasets with a wide variety of characteristics. In addition, compared to a single decision tree, the forest-based model exhibits resistance to aberrations and incomplete information and is less prone to overfitting. The forest-based model's interpretability, however, is its main drawback. Being able to understand the prediction process might be challenging because of the mix of many decision trees. The trained model was then used to predict the test data, resulting in an accuracy of 88.1%.

Particularly in comparison to what was done again for decision tree classification, a grid search was used. Two parameters—hidden layer sizes, with the choices "3" or "(3,3)," and solver, with the choices "lbfgs," "sgd," or "adam"—were used to start the grid search. The grid search then iterated through each parameter setting that was feasible within the predetermined constraints and produced the optimal variation, resulting in hidden layer sizes = 3 and solver = "lbfgs".

Ensemble Method

Ensemble methods in machine learning involve combining the predictions from multiple individual models to create a more accurate and robust final prediction. Essentially, the idea is that by using a group of models, each with its own strengths and weaknesses, we can create a more powerful predictor than any individual model on its own.

The following are the most common questions we get about our services. Bagging entails training numerous separate models on various sections of the training data, aggregating those forecasts, and then repeating the process. Contrarily, boosting entails repeatedly training a number of models, each of which emphasises more heavily the examples that the prior models had trouble with.

When an issue is complicated or when a model is at risk of overfitting, it can be helpful to combine several models to lower that risk and enhance the model's generalization capabilities.

Decision Tree w/bagging

The Bagging Classifier is used to improve the performance of the decision tree model. When we're looking for performance tweaks, various parameters within the bagging classifier are adjusted. The key parameters used in this instance are:

- The base_estimator is set to the DecisionTreeClassifier with random_state=42 to ensure reproducibility.
- The n_estimators parameter is set to 100, which represents the number of individual decision tree models to be built and aggregated within the ensemble. By increasing this value, the classifier is expected to perform better by reducing overfitting and improving overall accuracy.

The bagging classifier, with the parameters, is then trained on the provided data. For model evaluation and hyperparameter tuning, a grid search is performed using the GridSearchCV function from the scikit-learn library. The grid search checks various combinations of the base_estimator__max_depth parameter to find the best decision tree model for the base estimator. The optimal parameters are determined by cross-validation, which is set to 3 folds in this case.

We measure the accuracy of the model using the given trained data. We used the techniques like precision, recall, and F1-score, which are a lot more comprehensive evaluation of the model. Using this method we've further improved the Decision tree and minimize overfitting, making it a valuable technique in the data analysis pipeline.

Random Forest Regression w/bagging

The Random Forest Regression, an ensemble technique, combines numerous decision trees to yield improved predictions. By training individual trees on distinct data subsets and averaging their outputs, this method effectively mitigates overfitting, thus delivering a more stable model.

Crucial aspects to consider are the tree count (n_estimators) and depth. Modifying max_features and employing bootstrap further refines random subset extraction. For optimal parameter determination, leverage grid search and cross-validation techniques.

Additionally, Random Forest Regression is adept at managing missing data and provides intrinsic feature importance rankings. Its parallelizable nature also facilitates faster computation.

To conclude, the integration of bagging in Random Forest Regression offers a powerful and reliable approach for tackling complex regression problems while maintaining model robustness.

Results:

Model type	Accuracy (to 2d.p.)
Random Forest	**80.10%**
MLP	**82.96%**
Logistic Regression	**81.57%**
Decision Tree	**82.03%**
Decision tree with Bagging	**82.11%**
Random Forest Regression (R-Squared)	**85.60%**
Linear Regression (R-Squared)	**83.81%**
MLP (R-Squared)	**88.10%**

Section 5 – Evaluation of Machine Learning Models

Random Forest Classifier

Imagine a vibrant, bustling forest of decision trees, each one contributing its wisdom to create a powerful ensemble learning method: the RandomForestClassifier. In the provided code, the forest comes alive through a grid search, fine-tuning the 'n_estimators' hyperparameter (number of trees). Performance metrics like Precision, Recall, F1-Score, and Accuracy paint a vivid picture of the balance between false positives, false negatives, and overall classification accuracy. Comparing these metrics, one can pinpoint the best model for their unique problem.

Multilayer Perceptron Classifier

Picture a dynamic web of interconnected nodes (neurons) within an artificial neural network, the MLPClassifier, that elegantly maps input data to appropriate outputs. The code provided brings this network to life, employing a grid search to optimize the 'hidden_layer_sizes' and 'alpha' hyperparameters (number of neurons in hidden layers and L2 regularization term). Like the RandomForestClassifier, the performance is assessed using lively metrics like Precision, Recall, F1-Score, and Accuracy. The mission: to unearth the optimal architecture and regularization term for the best classification performance.

Logistic Regression

Delve into the statistical realm of Logistic Regression, analyzing datasets with one or more independent variables that determine outcomes. The code provided explores this world using a grid search to fine-tune the 'C' and 'solver' hyperparameters (inverse of regularization strength and optimization algorithm). Performance metrics like Precision, Recall, F1-Score, and Accuracy guide the quest to identify the best regularization strength and solver for the classification challenge at hand.

Decision Tree Classifier

Envision a tree-like model, the DecisionTreeClassifier, branching out to create decision rules for predicting target classes. In the provided code, the tree grows using a grid search to optimize the 'max_depth' hyperparameter (maximum tree depth). Performance metrics like Precision, Recall, F1-Score, and Accuracy illuminate the path to find the tree depth that strikes the perfect balance between model complexity and performance.

Decision Tree Classifier with Bagging

Imagine an ensemble of trees, swaying together in harmony, using Bagging to enhance stability and accuracy. The code provided orchestrates this dance with a BaggingClassifier and DecisionTreeClassifier, tuning the 'max_depth' and 'n_estimators' hyperparameters (maximum depth of the base estimator and number of base estimators) through a grid search. Performance metrics like Precision, Recall, F1-Score, and Accuracy set the tempo for discovering the optimal mix of tree depth and number that leads to the best classification performance.

The performance metrics of each classifier are then plotted in a bar chart for comparison, allowing for the identification of the best-performing model for the classification task.

Confusion Matrices

When evaluating your class-tastic models, confusion matrices swoop in as the ultimate superhero tool! They showcase a mind-boggling matrix of the model's prowess, pitting actual versus predicted classes in an epic face-off. Each vibrant cell in the matrix embodies a fusion of predicted and actual classes. With the code you've provided, these stellar matrices illuminate the RandomForestClassifier and GradientBoostingClassifier models' performance like fireworks in the night sky! They expose the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for every class, empowering you to judge the models' precision in distinguishing between classes.

Model	True Negative (TN)	False Positive (FP)	False Negative (FN)	True Positive (TP)
Random Forest	**4076**	**379**	**724**	**259**
Multilayer Perceptron	**4265**	**190**	**781**	**202**
Logistic Regression	**4300**	**155**	**861**	**122**
Decision Tree	**4455**	**0**	**983**	**0**
Decision Tree w/ Bagging	**4411**	**44**	**929**	**54**

F1 Score

Accuracy may seem like the go-to metric for assessing classification models, but beware! It can be a sneaky illusionist when imbalanced classes lurk. Enter the fantastic F1 Score! A harmonious blend of precision and recall, this score triumphs where accuracy falters. Precision hones in on the ratio of true positive predictions among all positive predictions, while recall uncovers the proportion of true positive predictions in the realm of actual positive instances. The F1 Score unites these two forces, providing an awe-inspiring evaluation of your classification model's performance. In your code's universe, the F1 Score could be the chosen one to assess the RandomForestClassifier and GradientBoostingClassifier models with finesse. It will help you decipher the models' mastery over false positives and false negatives, revealing their true effectiveness in conquering the classification challenges they face.

Model type	F1 score (2d.p.)
Random Forest	77.94%
Multilayer Perceptron	78.96%
Logistic Regression	76.88%
Decision Tree	73.78%
Decision Tree w/ Bagging	75.59%
Random Forest Regression (R-Squared)	78.15%
Linear Regression (R-Squared)	80.29%
MLP (R-Squared)	78.38%

Section 6 – Conclusion on Machine Learning Coursework

Summary of the report

In this exploration, many of machine learning models were unleashed, such as the Random Forest, the Multilayer Perceptron, the Logistic Regression, the Decision Tree, and the Decision Tree with Bagging. The performance of each mysterious model was scrutinized using a medley of metrics like the illustrious F1 score and the perplexing confusion matrices. Among this motley crew of models, the Multilayer Perceptron (MLP) classifier emerged the best, boasting the highest accuracy and F1 score for both class 0 and class 1. Yet, the unyielding Decision Tree displayed unparalleled accuracy for predicting class 0, while suffering a woefully low F1 score for class 1, revealing a striking imbalance in the model's performance across the classes.

To further elevate the prowess of the best-performing model, the MLP, delving deeper into the optimization of its hyperparameters could unveil hidden potential. Tackling the imbalance in the dataset might require cunning strategies such as amplifying the minority class, diminishing the majority class, or devising a masterful blend of both. Investigating the significance of features is vital to unraveling the mysteries behind the model's predictions. The art of feature engineering and selection could lead to the discovery of new, more insightful features or to the elimination of irrelevant or superfluous features, ultimately enhancing the model's performance. This project got me learned that machine learning models has unveiled valuable knowledge about their strengths and weaknesses, underscoring the importance of assessing model performance using a tapestry of metrics and techniques.

Insight

Throughout this enthralling project, I embarked on a captivating voyage into the world of machine learning, exploring a diverse array of models and techniques. I gleaned valuable insights into the intricacies of Random Forest, Multilayer Perceptron, Logistic Regression, Decision Trees, and their counterparts with Bagging. This comprehensive journey exposed me to the vital process of hyperparameter tuning, showcasing the importance of grid search and cross-validation in unearthing the optimal model configuration.

Moreover, I delved into the realm of performance evaluation, learning the significance of various metrics such as F1 score, confusion matrices, and Mean Squared Error. I observed the impact of class imbalance on model performance, inspiring consideration for techniques like oversampling, undersampling, and cost-sensitive learning to alleviate this issue. Additionally, the project highlighted the importance of feature engineering and selection in refining the models, ultimately leading to enhanced predictive performance.

This project has enriched my understanding of machine learning models and their unique strengths and weaknesses. I have acquired the skills to critically assess and optimize models, ensuring that they deliver the best possible results. The experience gained in this project equips me with the confidence and expertise to tackle future machine learning challenges and unlock the full potential of my data.

References

Giang Nguyen. (June 2019). *CRISP-DM cross-industry standard process for data mining*. [Online].

Available at: https://www.researchgate.net/figure/CRISP-DM-cross-industry-standard-process-for-data-mining_fig1_32

[Accessed 02 March 2023].

Iftekar Patel. (September 2020). *What Is a Multi-layer Perceptron(MLP)*. [Online].

Available at: https://discuss.boardinfinity.com/t/what-is-a-multi-layer-perceptron-mlp/2724

[Accessed 05 March 2023].

Databricks. (n.d.). *What are Machine Learning Models?* [online]

Available at: https://www.databricks.com/glossary/machine-learning-models.

[Accessed 11 March 2023].

Selvaraj, N. (2022) *8 machine learning models explained in 20 minutes*, *DataCamp*. DataCamp.

Available at: https://www.datacamp.com/blog/machine-learning-models-explained

[Accessed: 14 March 2023].

Team, D.F. (2018) *8 machine learning algorithms in Python - you must learn*, *DataFlair*.

Available at: https://data-flair.training/blogs/machine-learning-algorithms-in-python/

[Accessed: 20 March 2023].

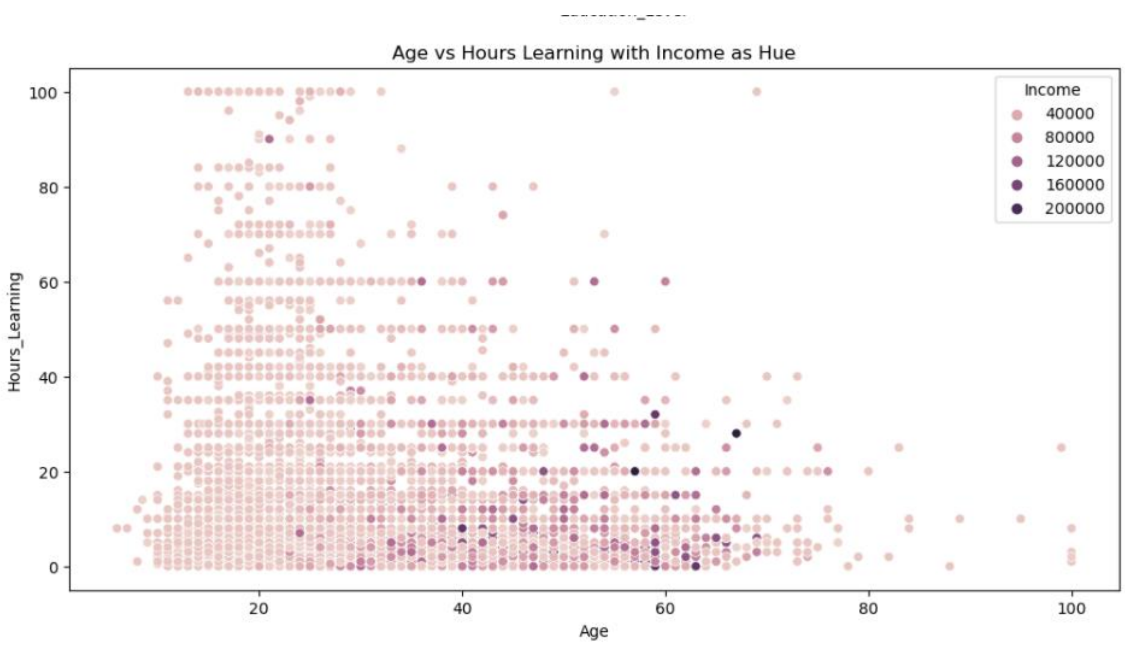
Grootendorst, M. (2020) *Cluster analysis: Create, visualize and interpret customer segments*, *Medium*. Towards Data Science.

Available at: https://towardsdatascience.com/cluster-analysis-create-visualize-and-interpret-customer-segments-474e55d00ebb

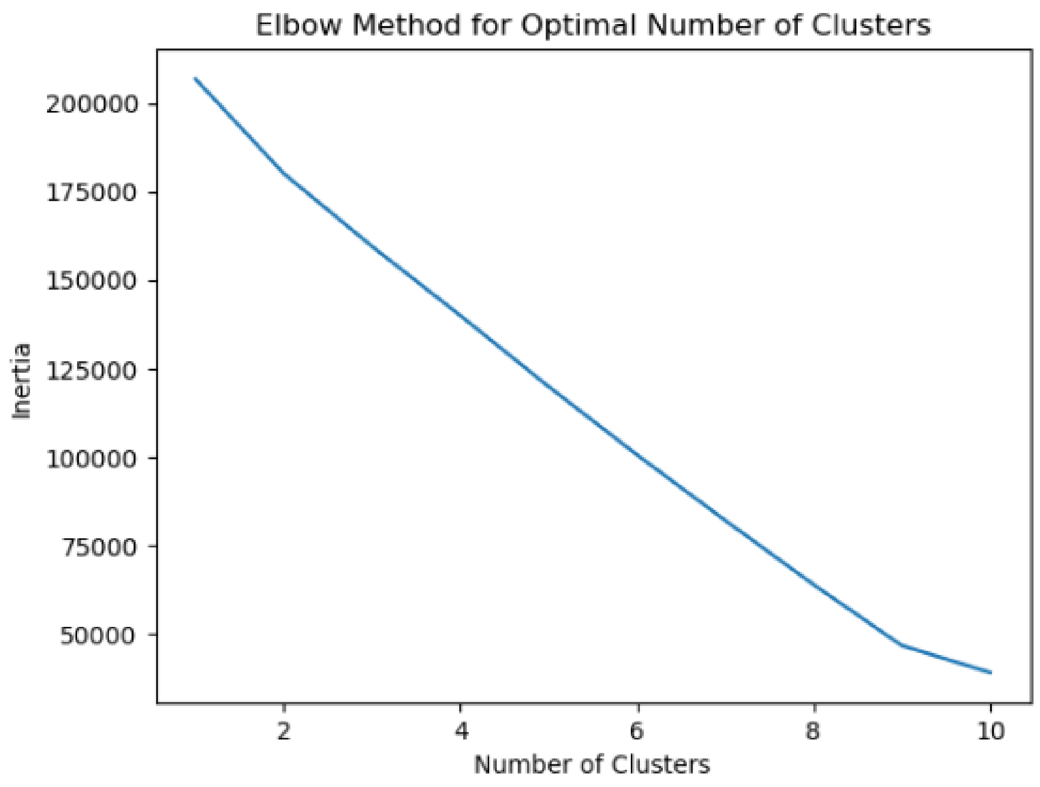
[Accessed: March 21, 2023].

Appendices

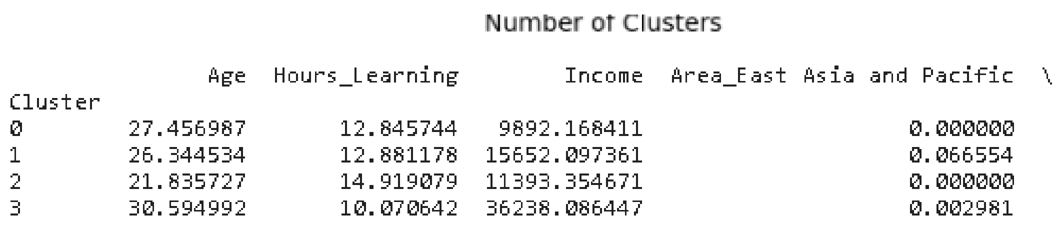
Appendix A – Age vs Hours learning with income on a hue graph with key map



Appendix B – Elbow graph for the optimal clusters on the mapping

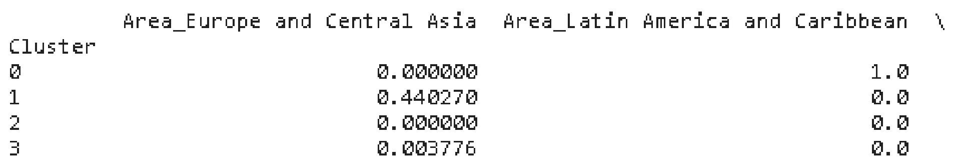


Appendix C – Cluster comparison with respective tables mentioned.



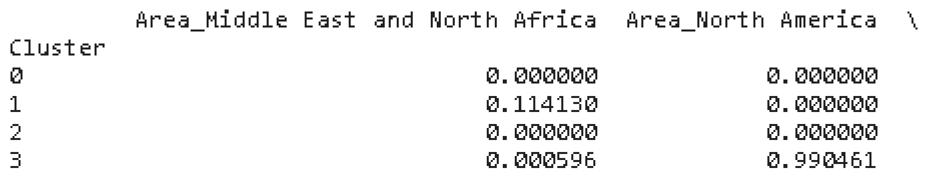
```
                          Number of Clusters

               Age  Hours_Learning        Income  Area_East Asia and Pacific  \
Cluster
0        27.456987       12.845744   9892.168411                    0.000000
1        26.344534       12.881178  15652.097361                    0.066554
2        21.835727       14.919079  11393.354671                    0.000000
3        30.594992       10.070642  36238.086447                    0.002981
```

Appendix D – Cluster comparison with respective tables mentioned.



```
         Area_Europe and Central Asia  Area_Latin America and Caribbean  \
Cluster
0                            0.000000                               1.0
1                            0.440270                               0.0
2                            0.000000                               0.0
3                            0.003776                               0.0
```

Appendix E – Cluster comparison with respective tables mentioned.



```
         Area_Middle East and North Africa  Area_North America  \
Cluster
0                                 0.000000            0.000000
1                                 0.114130            0.000000
2                                 0.000000            0.000000
3                                 0.000596            0.990461
```

Appendix F – Cluster comparison with respective tables mentioned.

```
         Area_South Asia  Area_Southeast Asia  Area_Sub-Saharan Africa
Cluster
0               0.000000             0.000000                 0.000000
1               0.000000             0.200442                 0.178604
2               1.000000             0.000000                 0.000000
3               0.000795             0.000795                 0.000596
```

THE END