NOTTINGHAM TRENT UNIVERSITY
School of Science and Technology

COMP20121

MACHINE LEARNING FOR DATA ANALYSIS

Using Machine Learning to Predict Developer's Income

by

Jamie Voce

N0868446

Contents:

Contents

Figures:

Tables:

Introduction:

What is CRISP-DM?

CRISP-DM (CRoss-Industry Standard Process for Data Mining) is a methodology used for data mining and machine learning first published in 1999. Due to CRISP-DM being a cross-industry standard, it can be implemented into many different data analysis projects irrespective of their domain (Great Learning Team, 2020). Because of this, it has many different applications within the data science sector and can be applied to research in a wide range of fields including medical research (Contreras, Yudith et al., 2018), analysis of the banking sector to help prevent company bankruptcies (Biceková, A. and Pusztová, Ľ., 2019) and even projects as obscure as predicting the tear strengths of woven fabrics (Ribeiro, R. Et al, 2020). The applications of data analysis and CRISP-DM are of particular importance to the business and finance sectors where analysis can be conducted to forecast stock prices, improve cross-sales and perform product performance analysis (Bose, I. and Mahapatra, R.K., 2001). CRISP-DM can be split into six phases as shown in Figure 1.

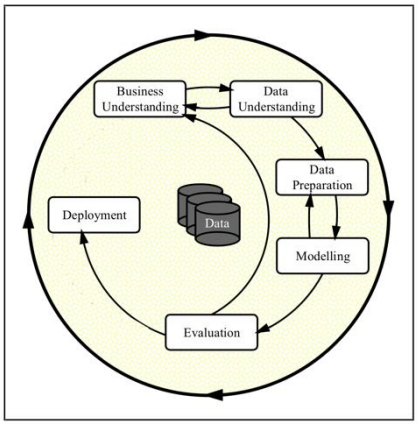


Figure 1: The CRISP-DM process (Writh, R. and Hipp, J. 2000)

Stage 1 - Business understanding:
The first stage of CRISP-DM focuses on viewing the task from a business perspective. It helps uncover success criteria and makes you ask what are the objectives and requirements of this project and what will the impact of the intended results be on the company or area it is being conducted? (Shafique, U. and Qaiser, H., 2014)

Stage 2 - Data understanding:
This stage is about familiarising yourself with the data. Learning what each attribute represents as well as the type of data it holds. At this stage it is also useful to examine the quality of data, making sure to identify outliers and conflicting instances.

Stage 3 - Data preparation:
This stage involves preparing the data, so it is ready to be used for modelling, this includes but is not limited to:
choosing which columns to use as features and which to use as the target variable,
cleaning data to remove outliers or empty cells or even constructing new attributes based on other existing attributes. Further data preparation includes encoding, data transformation and data normalisation.

Stage 4 -Data modelling:
The first step of the data modelling stage is to decide upon which classification methods will be used if they're not already chosen during the business understanding stage. This stage also involves hyper-parameter tuning to optimise models to achieve greater performance.

Stage 5 - Evaluation:
During the evaluation, it is important to report on the performance of the different models used so you can gain a better understanding of where the implemented models fail and how they can be improved. This can be done by measuring accuracy, precision, recall or the F1 score. Visual evaluation methods can be used such as confusion matrices and ROC curves.

Stage 6 - Deployment:
The final stage of CRISP-DM is where the results from the models are used to inform business change. This refers to stage 1 where the objectives of the project were uncovered.

Data analytic task and how I am applying CRISP-DM?

My task is to create a machine learning algorithm that can predict whether salaries will be above or below the median salary of stack overflow users, determined by stack overflow's annual developer survey. I will do this by implementing multiple classification methods such as k-nearest neighbour, decision trees and logistic regression to find the solution which provides the highest degree of accuracy.

I will apply the CRISP-DM methodology to this task by going through the required stages. During the data understanding stage, I will make sure to understand each columns data type as well as the mean and standard deviation of columns that store numeric values. As part of the data preparation stage I will make sure to identify outliers using a variety of techniques such as box-plots, histograms, and scatterplots, making sure to then remove or replace these outliers once found. After this has been done, I will split the dataset into features and target variables, exploring the relationship between features and target variables using stacked bar plots for example.

The insight I intend to gain.

From this analysis, I intend to gain insight into the relationship between salary and a variety of attributes such as country, education level, years of professional coding experience and development field. This insight could be helpful to students and recent graduates who could gain a better understanding of their potential earnings and allow them to make a more informed decision about which career they would like to choose for themselves. As well as this, they could learn how their salary would increase over time based on their level of experience. On the other hand, employers could use this data to offer competitive salaries that entice more applicants while not offering excessively high salaries that are not in line with the rest of the industry.

Data Understanding, Data Pre-processing, Exploratory Data Analysis:

This data came from the 2020 Stack Overflow annual developer survey. These surveys are consistently the largest surveys of coders in the world. However, this year the survey aimed to be more diverse instead of bigger. This means that the data is representative of all types of coders regardless of age, ethnicity, sex, sexuality, or gender identity. The data was collected for various reasons including finding the most used programming languages, the highest paying programming careers and studying how representative the industry is as well as how respondents used Stack Overflow.

The data set has 64,461 rows and 61 columns, many of which are not required for predicting salaries such as SOAccount which is a column about whether the respondent had a Stack Overflow account and SurveyLength which recorded how the respondent felt about the length of the survey. My first task is to identify columns that are not needed for predicting salaries and dropping them from the data frame. The columns I am left with are shown in the table below.

Necessary columns:

Table 1: Analysis table of kept attributes.

Column name	Description	Data type	Number of Missing values
MainBranch (Categorical)	Is the respondent a developer by profession, a student, someone who sometimes codes as part of their work or an ex-developer etc.	Object	299
Age (Numeric)	The respondents age.	Float64	19015
ConvertedComp (Numeric)	The wage of the respondent converted into annual income in US dollars.	Float64	29705
Country (Categorical)	What is the respondent's country of origin?	Object	389
DatabaseWorkedWith (Categorical and container type)	Which database environments has the respondent done extensive development work in over the past year.	Object	14924
DevType (Categorical and container type)	The type of developer the respondent is e.g., front end, back end, desktop, or enterprise applications etc.	Object	15091
EdLevel (Ordinal)	The respondent's highest level of formal education.	Object	7030
Employment (Categorical)	The respondent's current employment status.	Object	607
Ethnicity (Categorical and container type)	The respondent's ethnicity.	Object	18513
Gender (Categorical)	The respondents gender.	Object	13904
LanguageWorkedWith (Categorical and container type)	The programming, scripting, or mark-up languages the respondent has worked with extensively over the past year.	Object	7083
MiscTechWorkedWith (Categorical and container type)	Other frameworks, libraries, and tools the respondent has worked with extensively over the past year.	Object	24147
NEWCollabToolsWorkedWith (Categorical and container type)	The collaboration tools the respondent has worked with extensively over the past year.	Object	11578
NEWOvertime (Ordinal)	How often the respondent works overtime.	Object	21230
OrgSize (Ordinal)	Approximately how many people work for the company the respondent worked for at the time they took the survey.	Object	20127
PlatformWorkedWith	The platforms the respondent has worked with extensively over the past year.	Object	10618

Sexuality (Categorical)	The respondent's sexuality.	Object	20469
Trans (Categorical)	Does the respondent identify as transgender?	Object	15116
UndergradMajor (Categorical)	The respondent's primary field of study as an undergraduate.	Object	13466
WebFrameWorkedWith (Categorical and container type)	The web frameworks the respondent has worked with extensively over the past year.	Object	22182
WorkWeekHrs (Numeric)	On average, how many hours per week the respondent works.	Float64	23310
YearsCodePro (Categorical)	How many years the respondent has been coding professionally, excluding education.	Object	18112
YearsCode (Categorical)	How many years the respondent has been coding for.	Object	6777
JobSat (Categorical)	How satisfied the respondent is with their job.	Object	19267

Conflicting instances:

Many columns had conflicting data. This included rows in which the years of professional or nonprofessional coding experience was greater than that of the respondent's age. In cases such as this, it is unwise to replace the data manually as there would at least be two conflicting attributes and replacing both age and years of coding experience, either professional or unprofessional without being able to trust that the other attributes are also not anomalous could cause inaccuracies. Because of this, I opted to drop rows where these conflicting instances occurred.

One other conflicting instance occurs when a respondent has spent more years coding professionally than they have spent coding in total. Replacing these values with a more suitable value proves to be challenging however as it is impossible to know which value is the anomaly. Has the respondent given a value for years of coding experience that is too high or a value for years of professional coding experience that is too low? Because this question cannot be determined, all cases where this conflicting instance occurred were dropped from the data frame.

Data analysis to handle anomalies and missing data:

Many categorical columns had missing values that could not accurately be inferred, in this case, I replaced missing values with the value "unknown". This would later allow for increased accuracy as opposed to replacing data with the modal value for each attribute as in some cases inferring missing data could significantly impact a result. For example, inferring a respondent's education level based on their age and years of professional coding experience is possible however, if wrong, can lead to a decrease in overall accuracy as education level plays a significant role in determining a respondent's salary.

I grouped all the variations of non-binary gender's so I could uncover the percentage of respondents who were non-binary. I used this same method for the sexuality attribute, grouping all sexualities that weren't heterosexual, homosexual, or bisexual together to form another category called other.

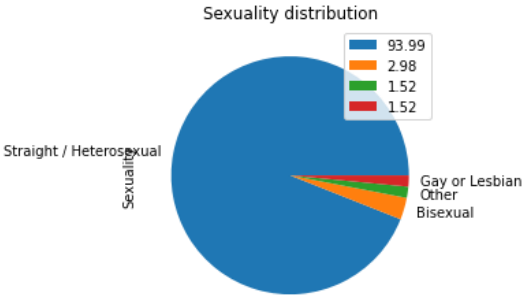


Figure 2: A pie chart to show the distribution of Sexuality in the data set.

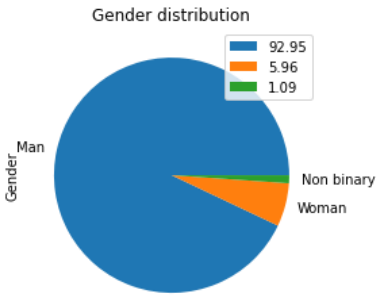


Figure 3: A pie chart to show the distribution of gender in the data set.

Figure 2 show that almost 94% of respondents identified as heterosexual and, as shown in Figure 3, roughly 93% of people in the data set identified as male. Surprisingly, this also revealed that 6.01% of respondents were not

heterosexual which meant that the LGBT community outnumbered women in the industry. Once we look at the distribution of gender and sexuality in the US alone **(see figs 6 and 7 in the appendix)** we can see that the gender and sexuality gap begin to shrink. This shows how computer science related fields are typically male-dominated even in the most well-developed countries however countries such as the united states have more women entering the discipline.

Based on the results found in both Figure 2 and Figure 3, I concluded that globally, it was fair to expect that missing values in the gender column would most likely be male and that missing values in the sexuality column would most likely be Straight/ heterosexual.

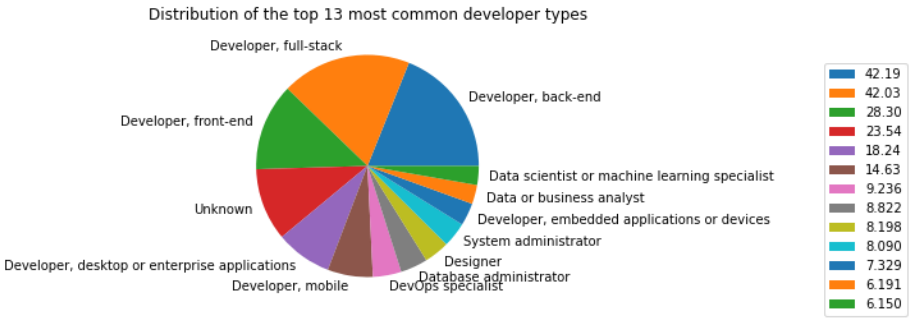


Figure 4: Distribution of the top 13 most common developer types.

As the 'DevType' attribute had multiple jobs listed in a column, I split each job into an unique binary column. As developers listed multiple developer types, the percentages seen in Figure 4 do not add up to 100%. Only the top thirteen developer types are shown in this pie plot to make it more readable however there were twenty-four unique developer types in the dataset. Doing this allowed me to uncover the distribution of job titles. Not including unknown values, the three most common dev types are back-end developer, full-stack developer, and front-end developer.

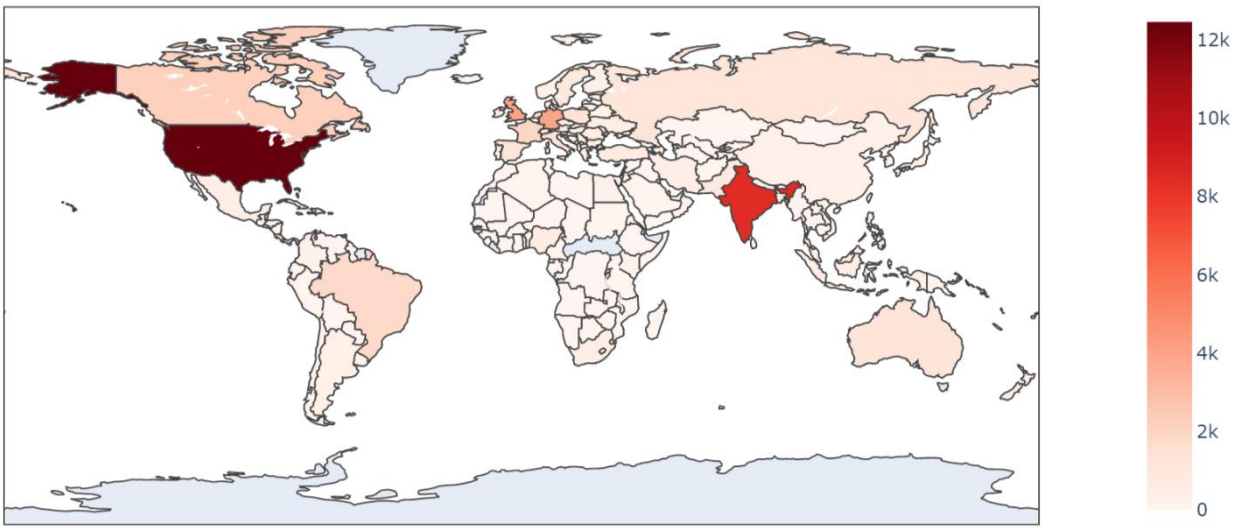


Figure 5: A choropleth map to show the most common countries in the data set.

The choropleth map shown in Figure 5 was created to analyse the country attribute and visualise which areas appeared most frequently in the data set. This showed that the United States and India were the countries in the data set that were the most common and therefore would have the greatest impact on the median salary of the data set.

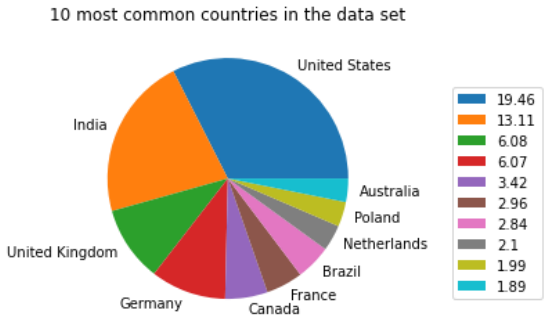


Figure 6: 10 most common countries in the data set.

To further analyse to country attribute Figure 6 was created to show the ten most common countries in the dataset along with each country's percentage frequency within the overall dataset. We see here that the United States, India, and the United Kingdom are the three most common countries in this dataset, accounting for over 35% of the dataset when combined. This means that if India's mean salary is significantly lower than the global mean salary, which I hypothesise that it will be, then the mean and median salaries globally will be much lower than the mean and median salaries in the United Kingdom and the United States.

Next, I needed to convert the data type of YearsCodePro from object to float64 so I could find the summary statistics of all the numerical attributes in my data set. I decided to round all values in the age and YearsCodePro columns to the nearest whole number, this was done because I knew I wanted to group the salaries by these attributes when it came to replacing the missing values. By rounding the values in these attributes, I would be increasing the accuracy of the mean calculated when performing the group by method as there would be more data in each group.

	Age	ConvertedComp	WorkWeekHrs	YearsCode	YearsCodePro
count	45446.000000	3.475600e+04	41151.000000	57684.000000	46349.000000
mean	30.834111	1.037561e+05	40.787028	12.709053	8.491812
std	9.585454	2.268853e+05	17.815656	9.717353	7.905296
min	1.000000	0.000000e+00	1.000000	0.000000	0.000000
25%	24.000000	2.464800e+04	40.000000	6.000000	3.000000
50%	29.000000	5.404900e+04	40.000000	10.000000	6.000000
75%	35.000000	9.500000e+04	44.000000	17.000000	12.000000
max	279.000000	2.000000e+06	475.000000	51.000000	51.000000

Figure 7: A table to show the summary statistics of the five numerical attributes.

Figure 7 shows that the attributes needed to be cleansed as the max-age was 279 and the max work week hours was 475 hours, both clear outliers.

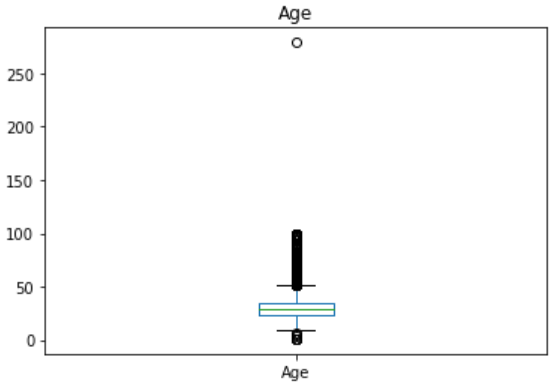


Figure 8: A boxplot to show age before data cleansing.

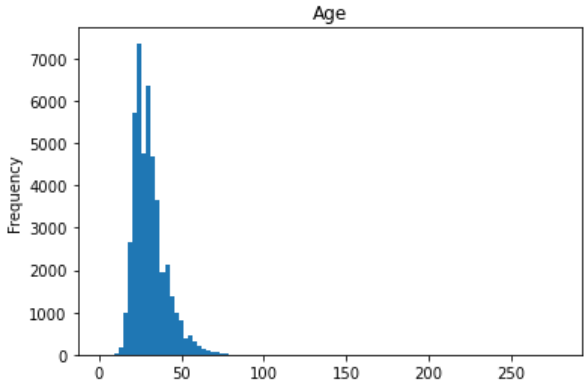


Figure 9: A histogram to show age before data cleansing.

Figure 8 and Figure 9 shows how the age attribute needed to be cleansed. As the boxplot showed clear outliers above and below the expected range. The histogram also showed how the values weren't distributed as expected, having the respondent's age go beyond that which could be considered likely. I decided that the best course of action was to limit the age to a maximum of 60 and a minimum of 16 as that would immediately remove the clear outliers. These values were chosen as it was unlikely anyone under the age of 16 would have a job relevant to computer science and they would also not be university students. The maximum age of 60 was chosen as it was unlikely a respondent would be that old and it was more than three standard deviations away from the mean.

The missing age values were replaced with the average age grouped by other attributes such as YearsCodePro, MainBranch, Country and EdLevel. I progressively lowered the number of other attributes in the group so that more missing values could be replaced. This meant that the data I was adding to the data frame was as accurate as possible.

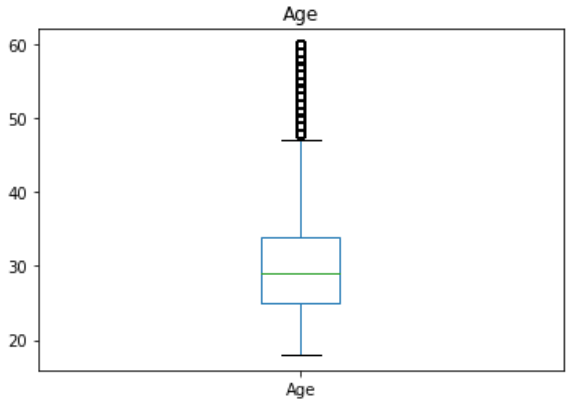


Figure 10: A boxplot to show age after data cleansing.

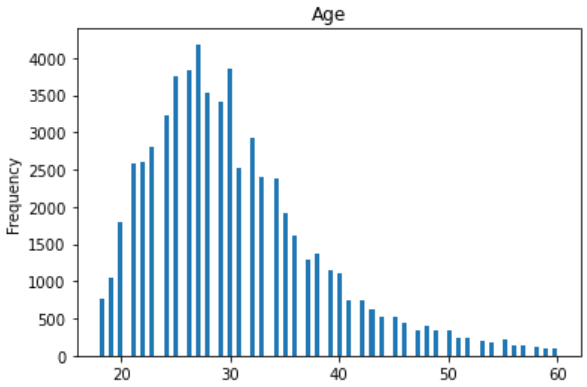


Figure 11: A histogram to show age after data cleansing.

Error! Reference source not found. shows some outliers however it is reasonable to assume that some people over the age of 48 will still work in the industry. **Error! Reference source not found.** shows a histogram with a bell curve with a mean age of 30, which is to be expected and thus can be considered cleansed.

Following this, I employed a similar system to cleansing the yearsCodePro attribute. I decided that since the minimum age in this dataset was now 18 and the maximum age was 60, I would limit the maximum value of yearsCodePro to 42 (the maximum age – the minimum age). This was chosen because if a respondent was to start work at the age of 16, they would have 42 years before they became the maximum age in the data set, 60. I then performed a series of group-by's to replace the missing data with the mean for their respective groups.

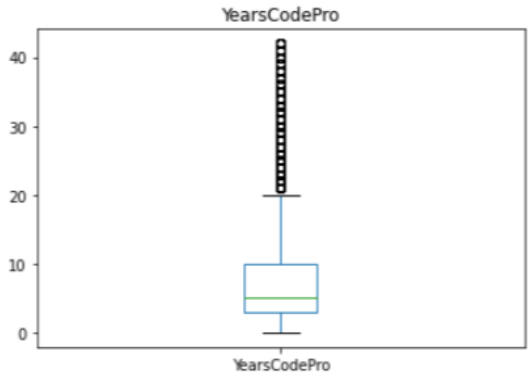


Figure 12: A boxplot of YearsCodePro after data cleansing.

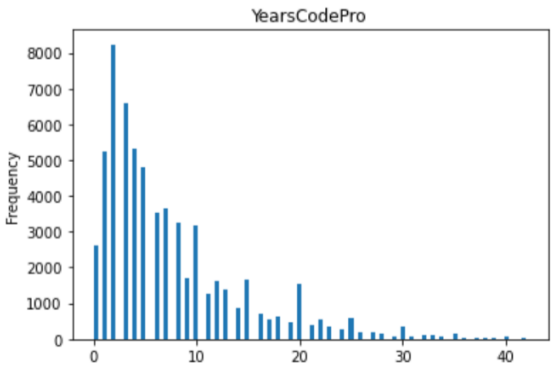


Figure 13: A histogram to show the distribution of YearsCodePro after data cleansing.

Figure 12 shows how the 'YearsCodePro' attribute seemingly still had many outliers however after careful consideration I concluded that this was not truly the case. It is fair to assume many respondents will work for longer than 21 years, as the age of retirement is greater than 44 (21+23 (The age most respondents would be upon receiving their bachelor's degree and thus the age most respondents would most likely start their professional careers)). Figure 13 shows that most respondents have a small amount of professional experience. This would make a lot of sense given that the computer science industry is rapidly growing and ergo, the number of young graduates with very little professional experience coming into the industry is much greater than the number of older employees leaving the industry.

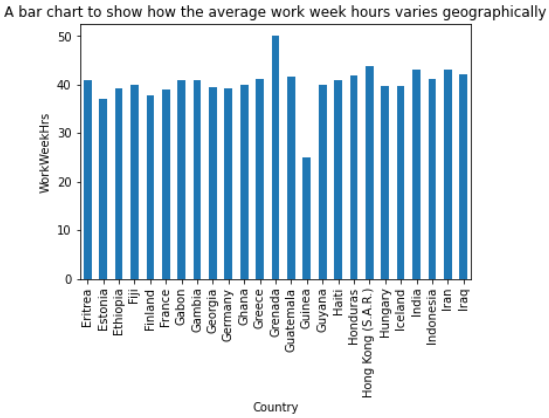


Figure 14: A bar chart to show how the average work week hours varies geographically.

For workweek hours I dropped all rows where respondents worked less than 16 or more than 58 hours per week. This is because they remove the extreme edge cases from the dataset where respondents have extremely short, or extremely long, work hours.

I chose to focus on replacing null values with data that was partly based on the respondent's country, this was because I noticed that country had a slight impact on workweek hours which could also go on to cause a significant difference in predicted salary. For example, the average respondent from India worked 43.2 hours per week, compared to the average UK respondent, who on average worked 39 hours per week.

I grouped rows by age, employment, country, and gender and then replaced the missing values with the mean workweek hours for other respondents who shared those four attributes.

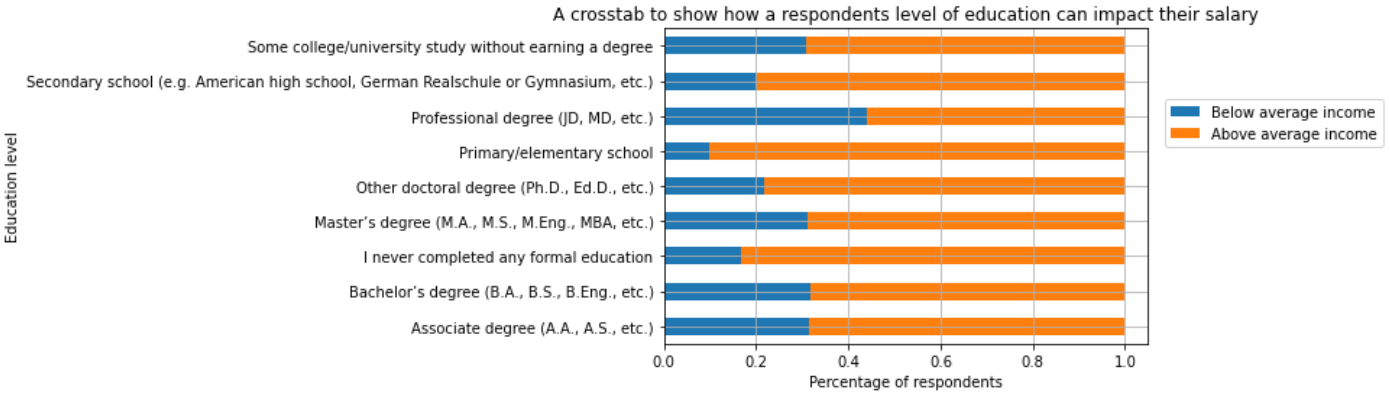


Figure 15: A cross-tab to show how a respondent's level of education can impact their salary

Figure 15 shows how there were significant outlier's to be found in the converted comp attribute. Before any form of data cleansing, respondents whose highest level of formal education was primary/ elementary school were more likely to earn greater than the median salary than respondents with any form of degree, including doctoral degrees.

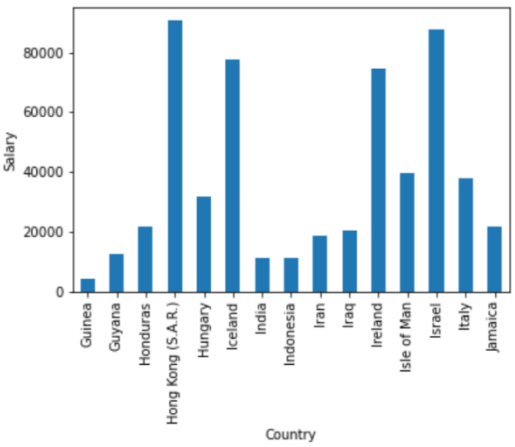
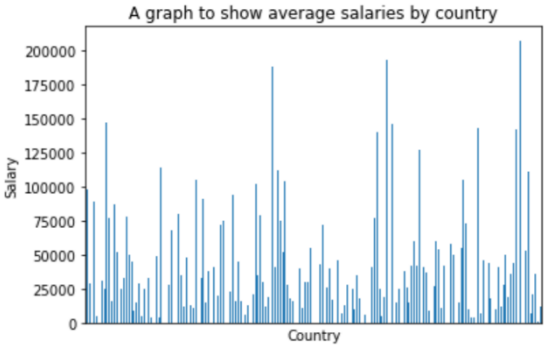


Figure 16: Mean salary against country.

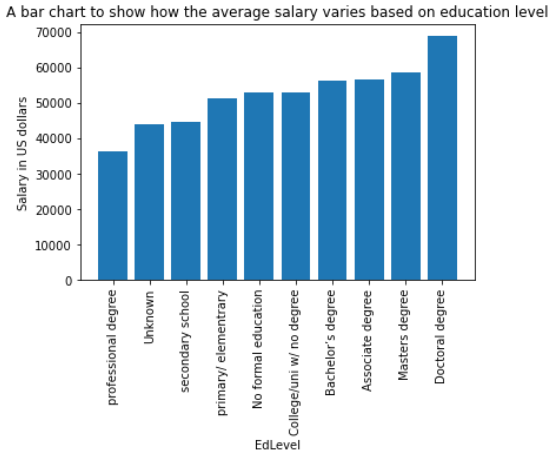
The importance of Figure 16 cannot be overstated. India is the second most frequent country in the stack overflow 2020 data set, with Indian respondents accounting for 13.11% of all respondents. The only country with more respondents was the United States with 19.46%. The implications of India having such a low mean salary shows how missing convertedComp values must be determined with the respondents country in mind. Salaries vary significantly based on the differing economic climates seen in various countries and, for this reason, replacing salaries with a mean value of all countries can cause disparities between the cleansed data and the actual lived experiences of the thousands of people from these less fortunate countries. This also shows that ConvertedComp outliers must be considered in relation to the country that the respondent is from. A salary of $90,000 would not be considered erroneous in the United States however would be considered erroneous in India.



Upon viewing the average salary for every country in the data set, as shown in Figure 17, we can see that some countries have salaries that appear to be erroneous. This seems to be because very few respondents come from certain countries and so, if the people in those countries are reporting high salaries, then it will raise that countries average salary significantly compared to other countries with more respondents. This doesn't necessarily mean that these countries have high average salaries, it just means there is a lack of data to suggest otherwise.

Figure 17: A graph to show mean salaries by country.

Because of these findings, I prioritised replacing missing 'ConvertedComp' values with the mean salary for that country however also grouped by other attributes for a more accurate result. These other attributes included WorkWeekHrs, Employment, EdLevel and YearsCodePro. Rows that were missing ConvertedComp, Country and WorkWeekHrs, were dropped as an accurate value for their salary could not be determined.



From Figure 18, we can see that as the education level increases, salary increases. More specifically we see that:
. On average, salary increases 3.95 % when respondents have a master's degree over a bachelor's degree.
. On average, salary increases 17.75% when respondents have a doctoral degree over a master's degree.
. And finally, the average salary increases by 23.40% when respondents have a doctoral degree over a bachelor's degree.

Figure 18: A bar chart to show how the average salary varies based on education level.

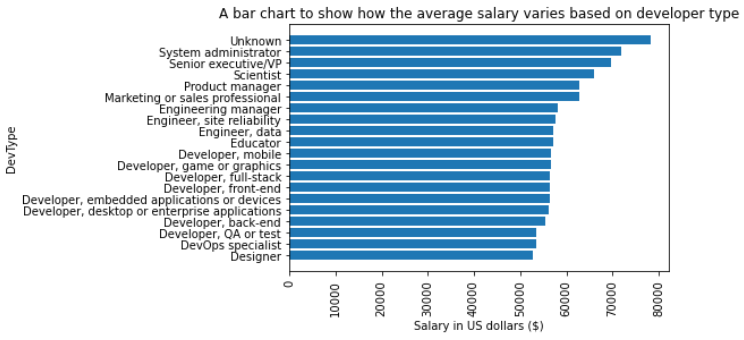


Figure 19 shows how salaries varied greatly based on the type of developer. As expected business roles such as managers, executives, and marketing were all in the top six highest paying developer types. Surprisingly, we see that system administrators, on average, are earning more than senior executives. While this could be true, it is more likely that the lack of senior executives taking the stack overflow survey means that the data for salary data for senior executives is not as accurate as it could be.

Figure 19: A bar chart to show how the average salary varies based on developer type.

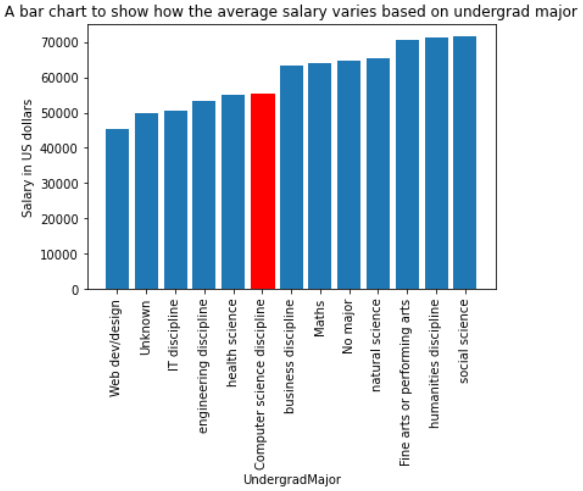


Figure 20: A bar chart to show how the average salary varies based on undergrad major.

Figure 20 shows how the average salary varies based on the respondent's undergraduate major. We see that more specialised disciplines such as business and mathematics have a higher salary than that of computer science related disciplines, which are highlighted in red. This is rather predictable given the types of technical skills these majors will need. For example, computer science will use a wide range of different skills with a wide range of salaries however business majors will typically need skills such as data analysis skills to further improve their business. Additionally, Figure 19 showed that business roles typically earned the highest salaries so it makes sense that if business related jobs were paying highly then those with business degree's would also be demanding higher salaries.

Figure 21: A graph to show the difference in salary for professional and non-professional experience.

Figure 21 shows how salary increases as you gain more experience. There appears to be a trend in this graph where more outliers become apparent as years of professional coding experience grows, this is to be expected given how there are fewer respondents in the data set with 25+ years of professional experience, meaning that outliers have a more significant impact on the data. Despite this however, we do see that overall, professional experience is valued more than nonprofessional experience.

Figure 22: A graph to show the difference in salary between men and women at each age group.

Figure 22 shows how the salaries for men and women rise at a very similar rate, however, shows that salaries vary much more for women than men. One likely cause of this would be that there are few women in this data set and therefore very few women in specific age groups, for this reason, salaries appear to very simply because there are fewer women in the data set. This would also explain why female's salaries appear to vary more later in life because this is the age group where we would expect to see the least women as historically computer science has been a male-dominated field.

<u>Removing anomalous salary data:</u>

To detect and appropriately remove anomalous salaries a series of equations needed to be created that limited the salary to a set range. This range would need to vary based on other attributes such as work week hours, years of professional coding experience, and age.

$$(c > 3000 * x)\ and\ (w < x)$$

The first equation used to remove anomalous salaries was created for work week hours. C is the converted comp value; w is the work week hours value and x is a for loop counter that iterates from 16 to 57. This equation is used to set the upper limit of a salary dependent on a respondent work week hours. As an example, a respondent who works

16 hours a week can earn a maximum value of $48,000 however someone who works 40 hours a week can earn a maximum of $120,000.

$$\big(c > 85000 + (5000 * x)\big)\ and\ (p < x)$$

The second equation was created to set the upper limit for a salary based on how many years of professional coding experience a respondent had. In this equation c was the converted comp, p was the number of years of professional coding experience the respondent had, and x was once again a for loop counter that iterates from 1 to 42. It is however important to note that an upper limit of $160,000 US dollars had already been placed on salaries.

$$(c > 3300 * x)\ and\ (a < x)$$

The final equation to set the upper limit for salaries is used to calculate which salaries are anomalous based on a respondent's age. This equation aims to refine the relationship between a respondent's salary (c) and their age (a). X is a for loop counter that iterates from 18 to 60, the range of possible age's in the data frame. E.g. $100,000 would be erroneous for a respondent who is only 23 years of age however it would be more likely that a respondent who is older at 40 years of age, could be demanding a salary that high.

Setting the lower limits for salaries in a way that was fair and did not artificially create relationships in the data frame proved to be challenging. In many countries, extremely low salaries would not be considered anomalous and hence the respondents country needed to be taken into consideration when attempting to infer when their salary is erroneous. To account for this a list of some of the richest countries in the data frame was created. To increase accuracy, countries with few respondents coming from there were not included in this list as it was not possible to infer if they were rich countries or the select few respondents who came from those countries were rich.

Rows are dropped from the data frame when the following equations return true, however are only used on respondents from the countries listed in Table 11 in the appendix.

$$(c < 300 * x)\ and\ (w > x - 1)\ and\ \ (w \leq x)$$

This equation, where c is the respondent's salary, w is work week hours and x is a for loop counter ranging from 16 to 57, was used to determine when a respondents salary was anomalous with respect to a respondent's work week hours.

$$(c < 833.33 * x)\ and\ (y > x - 1)\ and\ (y \leq x)$$

When setting the lower limits for salary in terms of its relationship with years of professional experience, it was important to create an equation that reduced extreme anomalies when the respondent had very few years of professional coding experience but also didn't remove valid salaries as respondents gained more professional experience. Because of this, a solution was made that started by only removing extremely low salaries but gradually increased its threshold, eventually allowing for more anomalies to be detected without ever unintentionally removing non-anomalous data. In this equation, c is the respondent's salary in US dollars, y is the respondents' years of professional experience and x is a for loop counter which ranges from 18 to 43.

$$(c < 600 * x)\ and\ (a > x - 1)\ and\ (a \leq x)$$

This equation was created to set a lower limit for salaries, in this equation c is the respondent's salary, a is their age, and x is a for loop counter that increments from 18 to 60. This aims to only remove extreme outliers but keep low salaries that have to potential to not be outliers. As an example, $10,000 for a 23-year-old in a developed country in most cases would be an erroneous salary however $15,000, while still low, may not be an outlier.

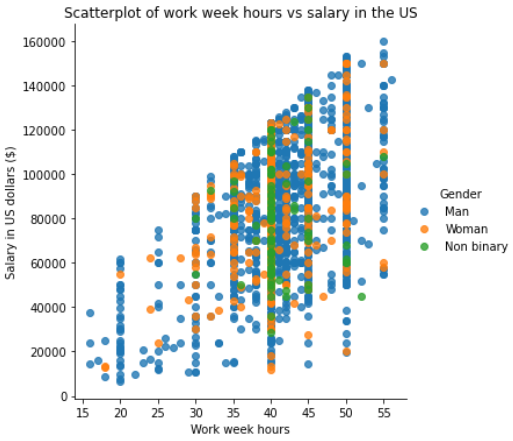


Figure 23: Scatterplot of work hours vs salary in the US.

After data cleansing had been completed on convertedComp, I produced the scatterplot seen in Figure 23, to see if the data showed a positive correlation between work week hours and salary. As you can see from the figure, most US salaries were earned by men who worked between 35 and 45 hours per week. Some respondents worked fewer hours and as predicted they earned a lower average salary, some people worked more hours per week and we can see in these cases that as work week hours increases, generally speaking, the average salary increases. I chose to look at one country in isolation for this scatter plot as the correlation would not be as apparent if all economies were displayed together. This graph shows a strong positive correlation between work week hours and salary for men and women in the US however it also shows a weak positive correlation for non-binary us citizens. This is most likely caused by the lack of data for non-binary individuals

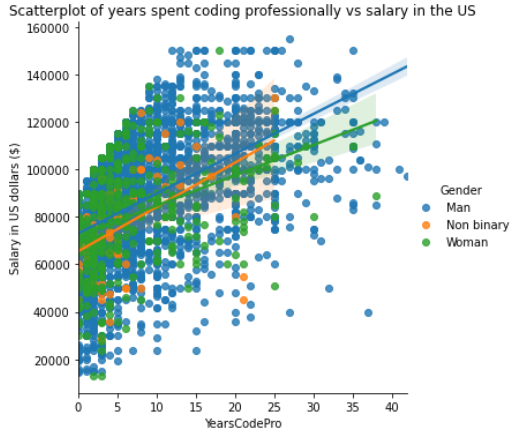


Figure 24: Scatterplot of years spent coding professionally vs salary in the US.

Figure 24 shows how in the US women's salary starts close to that of a man's salary. Over time, however, men's salaries increase at a faster rate than women's meaning that by the end of their career men are earning a significant amount more than their female counterparts. Another thing to note here is that the correlation between salary and years of professional coding experience is noticeably stronger for men than it is for women, it is unclear what causes this but it is likely caused by having more men in the data set than women.

The salaries of non-binary people saw a strong positive correlation between professional coding experience and salary and in fact, the data suggests that salaries for non-binary individuals increase at a faster rate than both men and women. This however is likely caused by a lack of data for non-binary individuals as we can see that there are very few non-binary respondents with ten to forty years of professional coding experience.

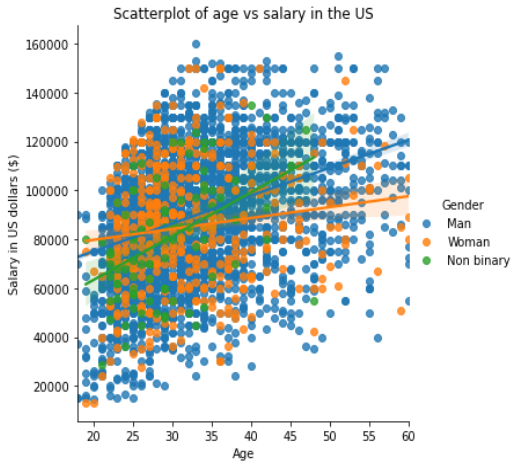


Figure 25: Scatterplot of age vs salary in the US.

Figure 25 shows that age has a clear relationship with a respondent's annual salary. As age increases, we see that salary tends to increase. There is a strong positive correlation for both men and women but once again we see that the correlation for men is much stronger than that of women, again it is believed that this is because of the lack of women in the data set, particularly older women. This scatterplot also provides a visual representation that shows most women in the industry are in the early stages of their career, typically being aged between 25 and 35. In the future, this means that as women age they will become more represented in the industry across all age groups.

Encoding data:
To encode the data multiple encoders were used. Ordinal data were encoded using the ordinal encoder and nominal data was encoded using the one-hot encoder. Making sure nominal data is encoded using the one-hot encoding is important as if nominal data was encoded using the ordinal encoder, the machine learning models could start to learn that the different values have an order to them when in reality, no order is present. This means using the correct encoder for nominal and ordinal data can increase the performance of the machine learning models.

Data that had multiple semi-colon separated values in one column were encoded using get dummies. Refer to (Table 1) for more detail on which attributes were ordinal, nominal and container types. This meant that data could be encoded more efficiently and achieve a higher degree of accuracy in the later machine learning models as there would be fewer columns and data wouldn't be duplicated. As an example, by splitting the attributes that utilise semi-colons as a delimiter, there is now just one column for the use of the Java language, however, if data were not split there would be multiple columns such as "Java", "Java; Python", "Python; Java", "Python; Java; C++", and so on. After the encoding process was complete, the shape of the data set was now 26,219, rows by 336 columns.

Data transformation and normalisation:
Some machine learning classifiers, namely neural networks and support vector machines require values to be normalised for the classifier to work. Encoded data values (x) must be normalised so that $-1 \leq x \leq 1$.

To transform and normalise the data, the standard scaler was used from sklearn. Using standard scaler sets the mean of each attribute to zero and the standard deviation to one. This ultimately increases the accuracy of the models by minimising the chance that the model will learn to create a bias towards certain attributes. As standard scalers can be influenced by outliers, it was important at this stage that outliers were removed where possible. Thankfully, many outliers were observed and appropriately handled during the data analysis stage. Only the training data was fit to this scaler as to not influence the test data and thus increase performance, however, both training and test data were transformed using standard scalar.

The standard scalar method uses the following equation to standardise each attribute in the data frame. This shows how standard scalar uses a normal distribution equation where $\boldsymbol{\mu}$ is equal to the mean vector of the attribute and $\boldsymbol{\sigma}$ is equal to the standard deviation of the attribute.

$$\boldsymbol{z = \frac{x - \mu}{\sigma}}$$

Cluster analysis:

The data was split into a low-income and a high-income data set. Low-income was defined as any salary less than or equal to the median salary and high-income was any salary greater than the median. Many different methods exist to uncover the optimal number of clusters, including k-means and 'the elbow method' (Alade, T., 2018). The elbow method was attempted for this task however the produced graph was less not adequate for the cluster analysis that needed to be performed. For this reason, multiple cluster sizes were tested to see the graphs they produced, then the optimal cluster size was chosen based on how appropriate the graphs were that they provided. The optimal cluster size was ultimately decided to be three. This was because two clusters did not split the data into clear groups and created a lot of overlap between the two groups and having more than three clusters proved problematic as it became difficult to analyse the key differences that made each cluster unique.

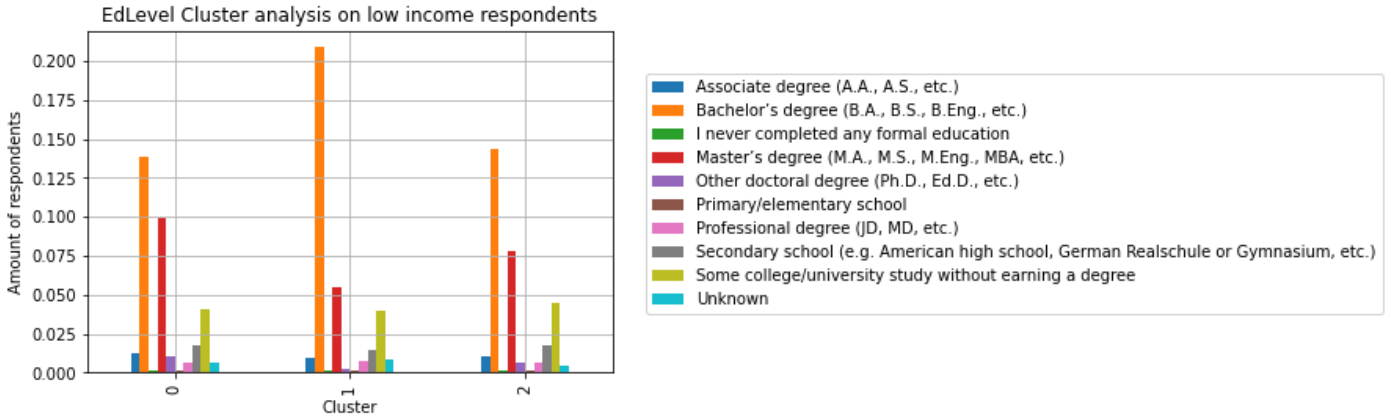


Figure 26: Education level cluster analysis on low-income respondents.

Figure 26 shows that in all three low-income clusters we see that bachelor's degrees and master's degrees are the two most common highest level of education achieved. Cluster 0 has the least bachelor's degrees but does have the most master's degrees and doctoral degrees however, in the case of doctoral degrees, they are still extremely rare. Cluster 1 has the most bachelor's degrees but the least master's degrees and doctoral degrees. Cluster 2 has more respondents with master's degrees than cluster 1 but less than cluster 0. It also has slightly more bachelor's degrees than cluster 0 but significantly less than cluster 1. All three clusters have similar amounts of respondents who have earned associate degrees or have no education past secondary school.

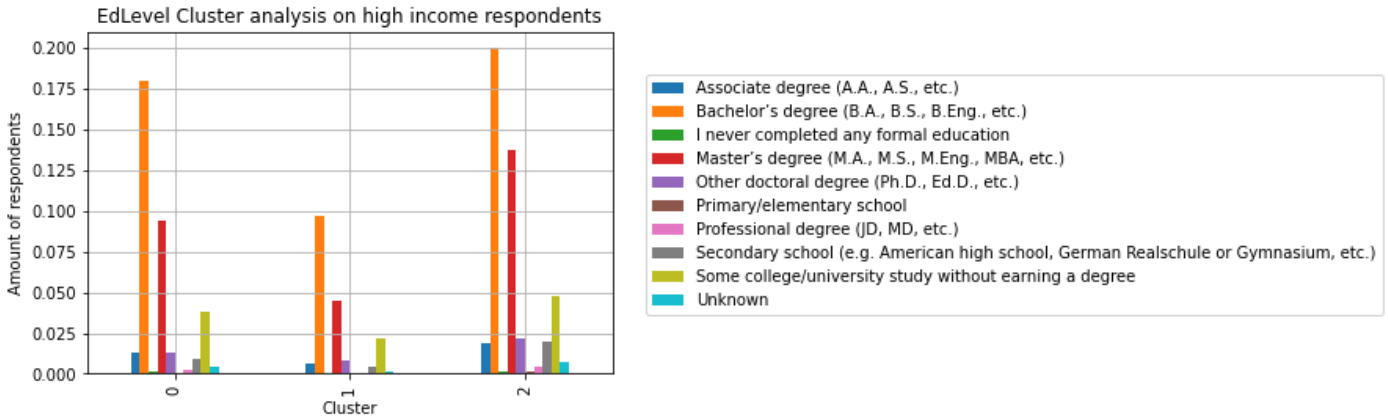


Figure 27: Education level cluster analysis on high-income respondents.

Figure 27 shows how bachelor's degrees were less common in the high-income clusters than in the low-income clusters however master's degrees were more common in the high-income clusters. Cluster 2 had the most bachelor's, master's and doctoral degrees and had the highest number of respondents who had some college or university study without earning a degree. Cluster 2 had slightly more respondents with doctoral degrees than cluster 0 however had over twice as many as cluster 1. Cluster 1 had the least amount of bachelor's, master's, and doctoral degrees. Cluster 0 had almost twice as many respondents with master's degrees than cluster 1 and had over 7.5% more respondents with bachelor's degrees than cluster 1.

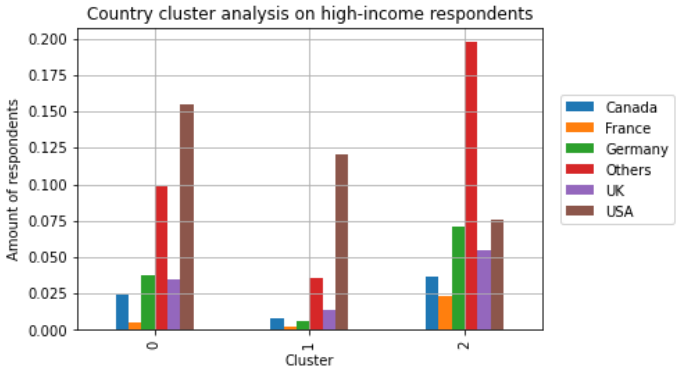
While most respondents from the low-income clusters (Figure 26) have degree-level education, bachelors, masters, and doctoral degrees are all less common in the low-income clusters compared to the high-income clusters (Figure 27). In fact, doctoral degrees are much rarer in the low-income clusters compared to the high-income clusters. We also see how more respondents in the low-income clusters have some college/university study but with no formal degree qualification.

As discovered in EDA, the country a respondent is from has a significant impact on salary and as such, you would expect the low-income and high-income clusters to show vast differences in their country attributes. I hypothesise that respondents with a high income will be from countries with a more stable economy whereas respondents from a low income will be from poorer countries. Specifically, I suspect India will appear much more frequently in the low-income clusters as opposed to the high-income clusters.

As expected, Figure 28 shows how the respondents country has a large impact on their income. Very few respondents in the low-income clusters came from the United States or the United Kingdom, two countries where the average salary is above the global average and it was common for respondents in these clusters to come from India, a country with an average salary that is significantly lower than the global average.

Figure 28: Country cluster analysis on low-income respondents.

In all three clusters, others were the most frequent country listed. Cluster 1 was almost entirely comprised of respondents from India and countries listed as 'others', with very few respondents also coming from Germany. It was also the cluster that had the most respondents from India, having more Indian respondents than cluster 0 and 2 combined. Cluster 0 had the most respondents from Canada, France, Germany, the UK, and the USA. Cluster 2 had some respondents from countries other than India and 'others' however still had India being the most common named country in the cluster.



Unsurprisingly, India was not as common in the high-income clusters (Figure 29). What was surprising however was that none of the respondents from the high-income clusters were from India. Canada, France, Germany, the UK, and the USA, were all much more common in the high-income clusters than the low-income clusters. This once again shows that the economic climate of a country can have significant effects on an individual's earning potential.

Figure 29: Country cluster analysis on high-income respondents.

Cluster 0 had the least number of respondents from all the countries listed in the legend and therefore was also the smallest cluster overall. Cluster 2 had the most respondents from countries listed as 'others' and had many respondents from Germany, the UK and the USA. Cluster 2 had the most respondents from the USA, having more respondents from the USA than cluster 0 and cluster 1 combined. Cluster 2 also had the most respondents from Canada, France, Germany, and the UK.

To perform cluster analysis on the undergraduate majors, computer science, computer engineering, and software engineering degrees were removed from the analysis because they were the most common degrees in every cluster, in both high-income and low-income developers. They were so common that they made it hard to analyse the frequency of the other majors as they were dwarfed in comparison to the computer science related disciplines.

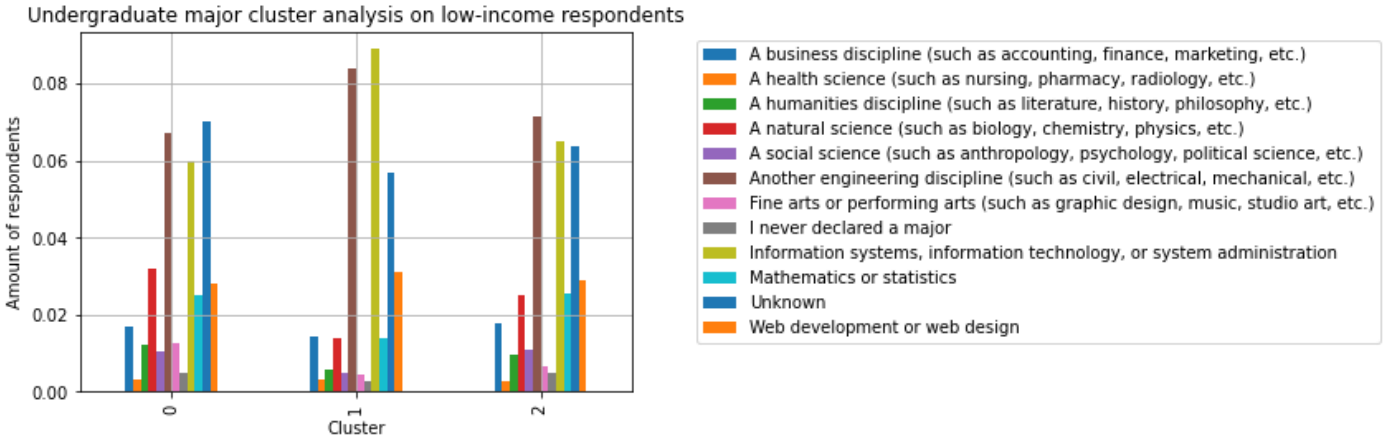


Figure 30: Undergraduate major cluster analysis on low-income respondents.

Figure 30 shows that undergraduate majors varied significantly in the low-income clusters. In cluster 0, engineering disciple degrees were the most common named undergraduate degree excluding computer science, followed by IT-related degrees. The least common degrees were health sciences and undeclared majors. Degree's listed as unknown were the most common undergraduate major in cluster 0. Cluster 1 again saw a significant number of respondents with engineering and IT degrees however, in this cluster, IT-related degrees were more common. Respondents who were missing an undergrad major however were far less common in this cluster than they were in cluster 0. Web development was more common in cluster 1 than cluster 0 and 2 however there were fewer respondents in cluster 1 with business, mathematics or statistics degrees than in cluster 0. Engineering degrees were the most common degree in cluster 2. IT degrees were also common, as were respondents who did not list their undergraduate major. Cluster 2 had the most respondents with mathematics or statics, web development/design or business degrees.

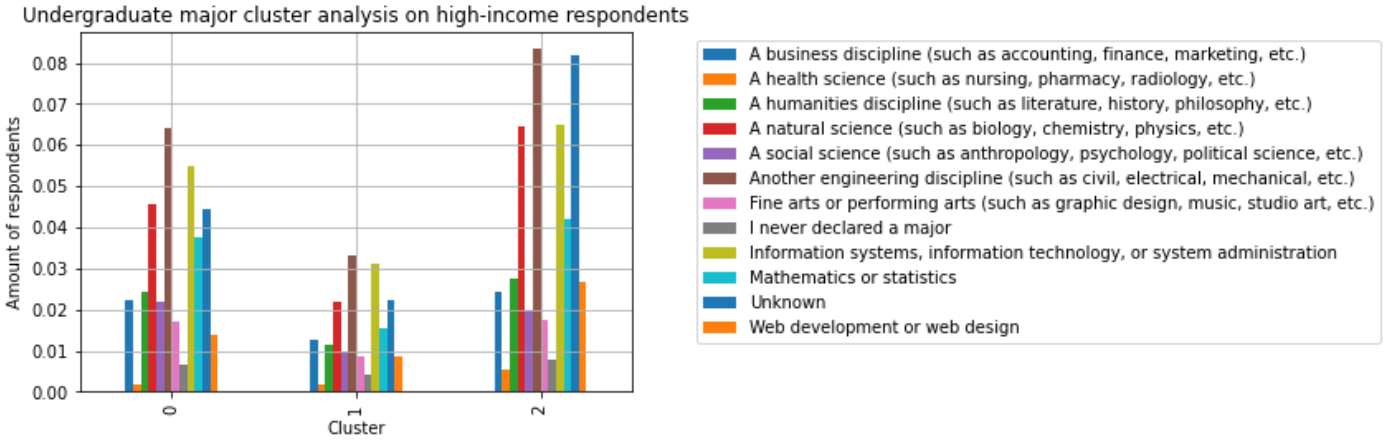


Figure 31: Undergraduate major cluster analysis on high-income respondents.

Figure 31 shows how IT-related degrees and engineering were still the most common degrees in the high-income clusters however other degrees were more common in these clusters. Health science was still extremely rare however natural sciences, social sciences and mathematics or statistics were all more common in the high-income clusters than the low-income clusters.

Cluster 0 was the cluster with the highest number of respondents with social science degrees. Engineering disciplines were the most common degree type and health sciences were the least common undergraduate major in this cluster. Cluster 1 was the smallest high-income cluster; it was like the other clusters in that engineering and IT majors were the most common undergraduate majors in the cluster and health science was the least common. Web development/ design was the second least common declared major. Cluster 2 was the largest cluster, engineering disciplines were the most common degrees in this cluster however respondents who were missing an undergraduate degree and were labelled 'Unknown' were a close second. IT, natural sciences and mathematics or statistics were

also exceptionally common in cluster 2. Once again, health sciences were the least common undergraduate major in this cluster.

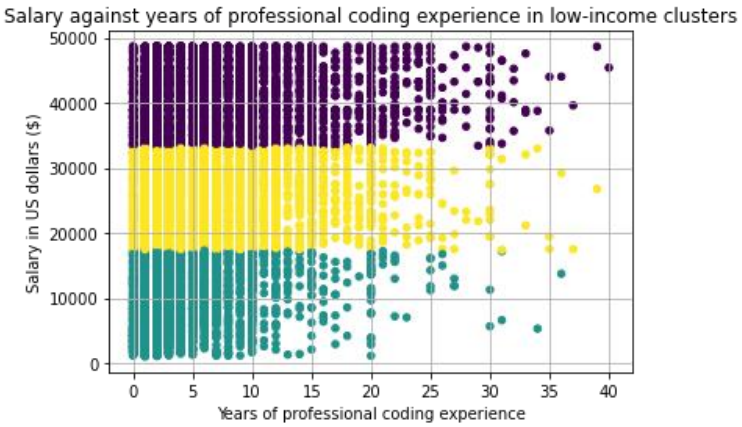


Figure 32: Salary against years of professional coding experience in low-income clusters.

Figure 32 shows the relationship between years of professional coding experience and salary on low-income developer clusters. Salaries ranged from around $0 to $50,000.

The blue cluster was the cluster with the lowest salaries, this cluster interestingly shows how respondents earning such a low salary typically had no more than ten years of professional coding experience. There were a few extreme cases where respondents in this cluster had over 30 years of professional coding experience which could be considered outliers; however, it would be hard to say with any certainty without also checking where these developers were from.

The yellow cluster was the cluster that had the average salaries for the low-income developers, this roughly ranged from $18,000 to $33,000. The typical years of professional coding experience was much higher in this cluster compared to the previous cluster at around 15 years of professional coding experience. The maximum years of professional coding experience was also higher in this cluster at 39 compared to 36 in the previous cluster.

The purple cluster was the cluster that had the wealthiest individuals from the low-income subsection of the data set. The maximum number of years of professional coding experienced usurpingly increased to 40 years over 38 in the previous cluster. What was surprising however was that the average years of professional coding experience did not seem to increase by any significant margin.

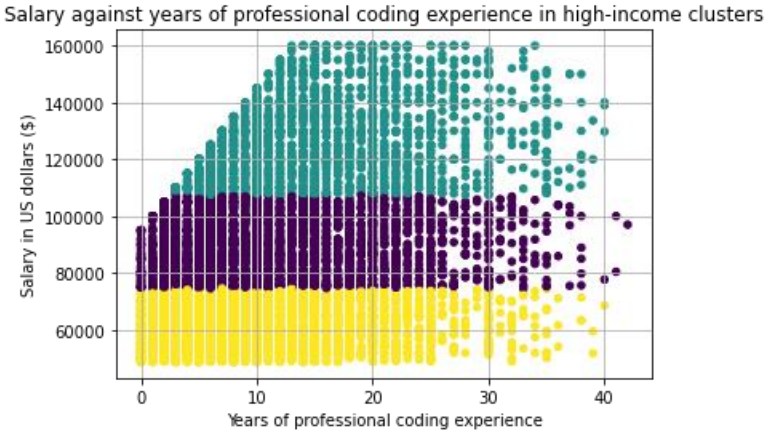


Figure 33: Salary against years of professional coding experience in high-income clusters.

Figure 33 shows the relationship between years of professional coding experience and salary on high-income developer clusters. Salaries ranged $50,000 to around $160,000. This was a much greater range than the low-income clusters.

The yellow cluster represents the poorest third of the high-income developers. They have more years of professional coding experience than the respondents in the low-income clusters and again, the maximum years of professional coding experience in this cluster is 40.

The purple cluster had salaries that were roughly in the range from $75,000 to $107,000. The maximum years of professional coding experience in this cluster was 42 years and the average years of professional coding experience showed a marginal increase when compared to the yellow cluster. The blue cluster was the cluster with the highest-earning respondents in the data set, their salaries ranged from around $107,000 to $160,000. This was by far the cluster with the largest salary range. The spread of respondents was also much wider in this cluster with some developers earning significantly more, while also having fewer years of professional coding experience. The average years of professional coding experience in this cluster seemed to be unchanged when compared to the yellow cluster however there was a decrease in maximum years of professional coding experience, with the max years of professional coding experience in this cluster being 40 years, compared to 42 and 40 years in the purple and yellow clusters respectively.

Machine learning for classification and their implementation:

Workflow of machine learning for classification:

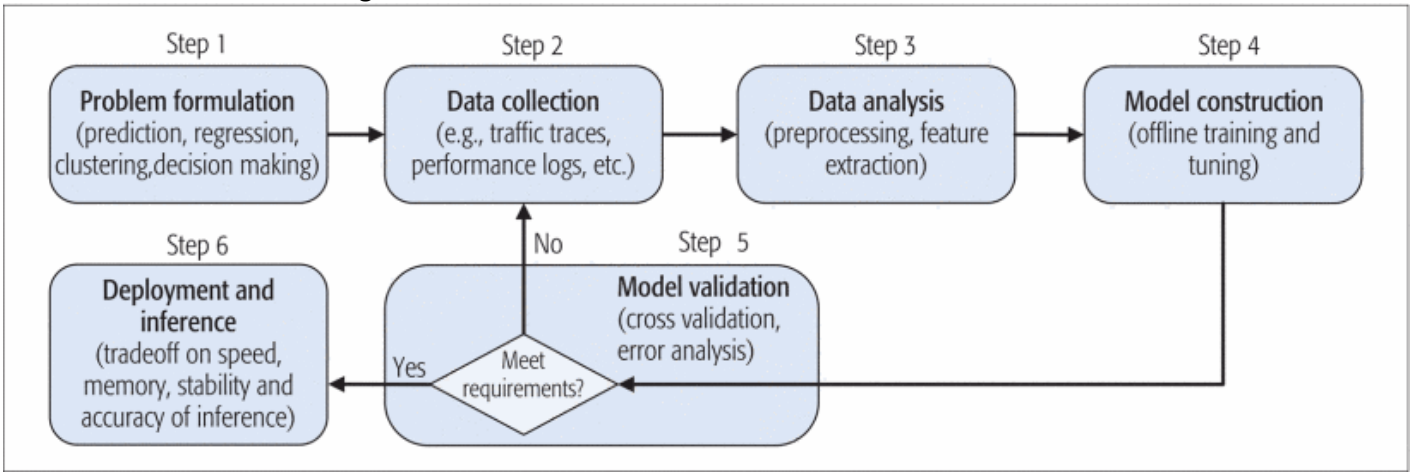


Figure 34: Flowchart of the machine learning process for classification (Wang, M. Et al., 2017).

Figure 34 shows the process used for machine learning for classification. At this stage, steps one to three were completed. The problem had been formulated with the task of creating a machine learning model that could predict when a developer would fall into the low-income or high-income bracket. Feature extraction and extensive data analysis had been performed, revealing conflicting instances and anomalies that had now been appropriately handled. Data pre-processing had been completed, encoding all attributes in the most appropriate manner by using a combination of ordinal and one hot encoding. The data had also been normalised as certain machine learning models, such as support vector machines, required data to be normalised before they could be used.

Model construction and model validation are closely related. Upon construction of a new model, the model's performance must be evaluated using a variety of performance metrics. This is so that model parameters can be chosen that maximise the performance of the models, this process is called hyper-parameter tuning. Methods such as grid search allow for hyper-parameter tuning to be handled automatically and can return the optimal parameters for a given model. One downside to hyper-parameter tuning is that it is a long process, especially when the base models have a long run-time and there are many possible combinations of hyper-parameters.

If a model's performance is still inadequate after hyper-parameter tuning, further data analysis will likely need to be performed to uncover outliers so that they can be removed from the dataset. This process is repeated until a model's performance is considered to be adequate.

Methods used:
To ensure the highest degree of accuracy was achieved multiple machine learning models were used. These methods were K-Nearest Neighbour, Decision Trees, Logistic Regression, Support Vector Machines, and an Artificial Neural Network using a Multi-Layered Perceptron. A Naïve Bayes classifier was also tested however after careful consideration it was considered to be unsuitable for this task. The artificial neural network classifier and support vector machine both required values to be normalised for the classifier to work. Encoded data values (x) must be normalised so that $-1 \leq x \leq 1$.

Hyper-parameter tuning was utilised to maximise the performance of the models where appropriate using the grid search method however this method was less suited to classifiers that had a long run time such as artificial neural networks and support vector machines. Additionally, I used ensemble methods to further tune performance by combining classifiers. The ensemble methods used were random forest, bagging, AdaBoost, and a majority voting classifier.

To be able to measure the effects of hyper-parameter tuning, non-hyper-parameter tuned versions of each of the base classifiers were created. This will be used to determine how successful the implemented hyper-parameter

tuning is. In this task, hyper-parameter tuning will be considered successful if the accuracy of the implemented model increases when compared to its non-hyper-tuned equivalent.

Logistic regression:
Logistic regression is often used for binary classification problems. It gets its name from the logistic function, also known as the sigmoid function. This function takes any real number and maps it into a value in the range from 0 to 1, excluding these limits (Brownlee, J., 2016), that represents the probability that the sample is positive or negative (Feng, J., Xu, H., Mannor, S. and Yan, S., 2014).

There were two hyper-parameters in the logistic regression classifier, the solver and penalty. The solver is the method used to implement the logistic function. Some solvers have longer run-times than others as they utilise more complex equations within their implementations. Two such solvers are Newton-cg and Saga. Saga was added as a possible value for solver in the parameter grid however it is mainly used for extremely large datasets and therefore is most likely not going to be the optimal solver for this model. The penalty parameter allows for the logistic regression model to be penalised for having too many variables in the model. This is also known as the regularisation parameter.

Possible hyper-parameter combinations are shown in the table below.

Table 2: Logistic regression hyper-paramters.

Solver	Sag
	Saga
	Liblinear
	Lbfgs
	Newton-cg
Penalty	L1
	L2
	Elastic-net

To select parameters for logistic regression I opted to use a parameter grid which determined which combination of given parameters achieved a higher degree of accuracy. The possible parameter options are listed in Table 2. Neither lbfgs or newton-cg work with a penalty of l1 however l1 and l2 were kept in the parameter grid so I could see which penalty worked best for the liblinear solver. Ultimately the best parameters were determined to be the liblinear solver paired with the l1 penalty.

A pipelining technique was then used which combined feature selection with classification. Feature selection was implemented using the sklearn method SelectFromModel and then the data was passed to the classifier created from the parameter grid. Feature selection can increase performance by removing unnecessary features from a data set, this also reduces the complexity of the produced machine learning model, thus decreasing run-time.

Other feature selection methods were tested including recursive feature elimination (RFE) and chi-squared.
RFE ranks predictors based on their importance. It recursively eliminates the least important predictors and re-fits the model until only the specified number of features remains (Brownlee, J., 2020). Chi-squared is used to test the independence of two events. In feature selection, it aims to select features that are highly dependent on the target variable (Gajawada, S., 2019), in this case, the income attribute.

Decision trees:
Decision trees work by creating a series of conditions for the data that each recursively split off, branch, into new nodes. Each branch eventually will lead to a leaf, this is the point at which one branch no longer splits. In a classification problem, this leaf node will then be used to ultimately decide on what to classify the respondent as (Gupta, P., 2017). In the case of predicting a developer's income, each leaf will decide whether to classify the income of a respondent as low-income or high-income.

One downside to decision trees is that they tend to lead to overfitting. The solution to this is called pruning, pruning is a technique that aims to identify redundant splits in the decision tree and remove them (Seif, George., 2018). This also reduces tree complexity which has the benefit of decreasing the decision tree's run time.

There are three hyper-parameters for decision tree classifiers, these are criterion, splitter and max depth. Criterion is the parameter used to measure the quality of a split. Possible criteria are gini and entropy. The entropy criteria splits nodes so that each split gives the most information gain whereas the Gini impurity splits nodes so that each split gives the least amount of impurity (Mithmrakumar, M., 2019).

Splitter defines how the decision tree chooses to split at each node. Possible splitters are best and random, the best splitter will use the best features to split whereas random will select features at random and split (Mithmrakumar, M., 2019). Unfortunately, the best splitter can often lead to overfitting. In these cases, the random splitter can be used.

The maximum depth defines how deep the branches of the decision tree can go. Small maximum depths can lead to underfitting while large maximum depths can lead to overfitting. Fortunately, thanks to decision tree's small run-time, identifying the optimal maximum depth can be easily achieved by using a parameter grid.

Table 3: Decision tree hyper-parameter tuning.

Criterion	Entropy
	Gini
Splitter	Best
	Random
Max_depth	Integer between the range of 15 and 30.

For the decision tree classifier, I again used a parameter grid to tune hyper parameters. I used this grid to determine which combination of criterion and splitter achieved the greatest accuracy. The possible parameters are shown in Table 3. As decision tree classifiers were much faster than other models such as K-nearest neighbour, support vector machines, and artificial neural networks, I chose to also use this parameter grid to identify the optimal max depth value between 15 and 30. The optimal hyper parameters determined by the parameter grid were criterion = gini, splitter = random and max_depth = 26.

Feature selection was once again used and combined with the optimal model using a pipeline.

Support Vector Machines:
Support vector machines (SVMs) work by identifying the optimum hyper-plane that separates target vectors on the opposing sides of the plane (Ragab, D.A., Sharkas, M., Marshall, S. and Ren, J., 2019).

SVMs have many hyper-parameters including, the kernel, the regularisation parameter (c) and gamma. The kernel takes data and transforms it into the required form. Possible kernels include linear, sigmoid and the Radial Basis Function (RBF).

Unfortunately, as SVMs have a long run-time, it was difficult to tune hyper-parameters. Because of this, the implemented support vector machine could be improved if further combinations of hyper-parameters were tested. The decision was made to not include the kernel hyper-parameter in the grid search when tuning hyper-parameters for this classifier. This was done to reduce run-time as reducing the number of combinations of parameters was particularly important; adding just one extra kernel to the grid search would almost double the run-time. The possible hyper-parameter values that were used in the parameter grid are shown in Table 4.

Table 4: Support vector hyper-paramters.

Parameter	Possible values	Chosen parameter
C	0.001	-
	0.01	-
	0.1	Chosen
Gamma	0.001	Chosen
	0.01	-

K-Nearest Neighbour:
K-nearest neighbour is one of the most widely used machine learning techniques for classification (Shouman, M., Turner, T. and Stocker, R., 2012). It works by identifying the nearest neighbours to a given query and use those neighbours to classify that query. The downside of using K-nearest neighbour classifiers is that they have poor run-time however the increase in computational power has made K-nearest neighbour more viable over time (Cunningham, P. and Delany, S.J., 2020.).

To uncover the optimal value for the number of neighbours (n), a graph was plotted with accuracy on the y-axis and with the number of neighbours on the x-axis as shown in **Error! Reference source not found.**. This graph allowed me t o determine at which value of n was accuracy at its greatest value.

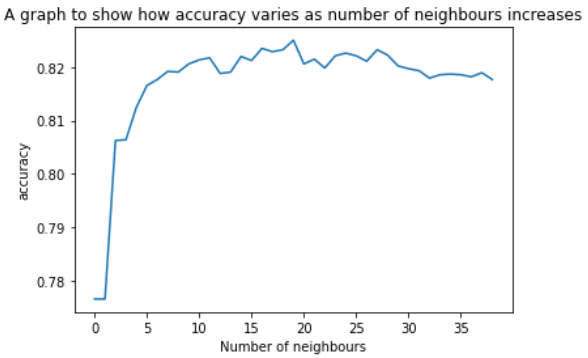


Figure 35: A graph to show how accuracy varies as the number of neighbours increases.

You can see how accuracy increased dramatically as the number of neighbours increased up until 9 neighbours, at which point accuracy started to become more varied. The final hyper parameters for the KNN classifier was determined to be as follows: weights = “distance”, metric = “euclidean” and number of neighbours = 17.

Artificial Neural Networks:
Artificial Neural Networks (ANNs) utilise a process known as back-propagation. This allows neural networks do adjust the weights based on the error rate of the previous epoch (Rungta K, 2020). If there is a large error rate, then weights will be adjusted by a large amount however if there is a low error rate then more minor adjustments will be made. This allows for ANNs to be dynamically adjusted and be fine-tuned to improve performance. For this reason, highly performant hyper-parameters must be selected for this model.

Hyper-parameter tuning for Artificial Neural Networks (ANNs) has a long run time which meant that running the grid search every time the notebook was executed was slow and unnecessary. To account for the long run time, I used a grid search to identify the optimal parameters but then created an ANN that used these parameters by default and didn’t require the grid search to be re-executed every time the code was run.

The activation parameter defines the activation function to be used. This tells the ANN how to process the input to make it produce the desired output. Possible activation functions include the tangent hyperbolic function (tanh), rectified linear unit (relu) and logistic.

Table 5: ANN activation function equations.

Activation function	equation
Tanh	$f(x) = \frac{2}{(1 + e^{-2x})}$
relu	$f(x) = \max x)$

Table 5 shows the activation function equations for both tanh and relu (Tiwari, S., 2020).

The possible solvers that were implemented in the parameter grid were Stochastic Gradient Descent (sgd), and adam, which is a stochastic gradient-based optimiser. The alpha parameter takes a given float value and uses it as a penalty, as hyper-parameter tuning for ANN's is slow, it was difficult to test many different values for alpha. The learning rate sets the schedule for updating the weights of the neurons. In this case, 'invscaling' stands for inverse scaling.

The ANN hyper parameters are shown in Table 6. The hidden layer sizes parameter sets how many hidden layers are used in the ANN and how many neurons are in each layer. It is important to select a hidden layer size that maximises accuracy without rapidly increasing run-time.

Table 6: ANN hyper-paramters.

Hidden_layer_sizes	(3)
	(3,2)
	(4)
Activation	tanh
	relu
	logistic
Solver	sgd
	adam
Alpha	0.0001
	0.05
Learning_rate	Constant
	Adaptive
	Invscaling

The optimal hyper-parameters were observed to be as follows: Hidden_layer_sizes = (3), activation = logistic, solver = sgd, alpha = 0.0001 and learning_rate = adaptive.

Deep Multi-Layered Perceptron (MLP):
Deep learning was also used to attempt to further the performance of the machine learning models. To implement deep learning, validation data sets had to be created to monitor the loss and accuracy of the model during training, this was not necessary for any of the other classifiers used. The historical information collected from training the deep MLP classifier will be used to measure the loss, accuracy, validation loss and validation accuracy.

Deep MLP's have two main hyper-parameters, epochs, and batches. The number of epochs is equal to the number of iterations that are made over the given data set. The Batches parameter is the number of samples from the data set that goes into each batch. The goal of hyper-parameter tuning for deep MLP's is to increase accuracy and decrease loss. One drawback to the deep MLP classifier is its tendency of overfitting, lowering the number of epochs is one way of preventing this. Other hyper-parameters include the number of layers and the number of neurons in those layers. To keep the model simple and generalised I opted to only use three layers. To further prevent overfitting, I used two layers with the relu activation, both of which with only eight neurons and a final layer using the sigmoid activation with only one neuron. I also used L2 penalty using the Keras regularizers method.

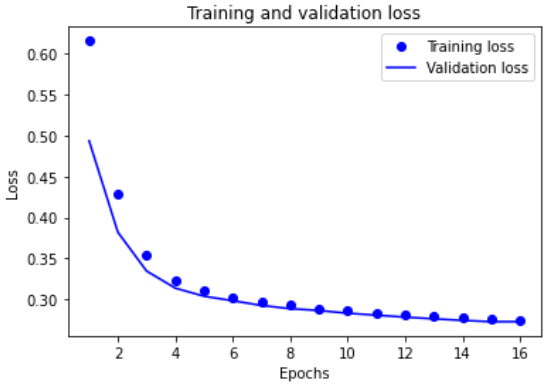


Figure 36: Training and validation loss for the implemented deep MLP.

Figure 34 shows how training and validation loss decreased at a similar rate as the number of epochs increased in the deep MLP. This shows how overfitting was successfully avoided as if overfitting. The training loss decreases to 0.26666, this could potentially be improved if further combinations of epochs and batch size were tested.

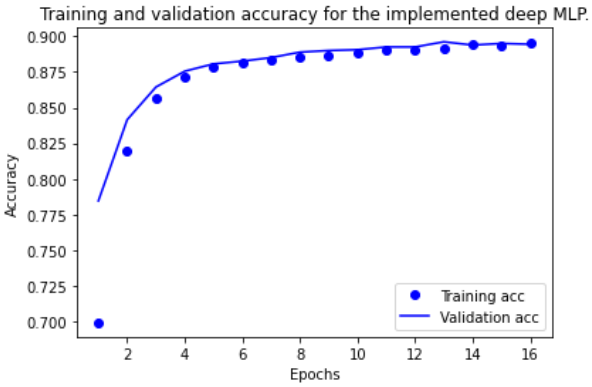


Figure 37: Training and validation accuracy for the implemented deep MLP.

Figure 35 shows how training and validation accuracy increased at a similar rate as the number of epochs increased in the deep MLP. This provides further evidence that overfitting was avoided. The training accuracy increases to a maximum value of 0.8983.

Ensemble methods:

Ensemble learning is the process of combining classifiers to increase the predictive performance of using singular models (Sagi, O. and Rokach, L., 2018.). One benefit of ensemble learning is that ensemble methods do not need to be hyper-tuned like single classifiers do. This is because, as they combine multiple classifiers, they often reduce error by averaging out results across the classifiers that comprise them.

Bagging:

Bagging, also known as bootstrap aggregating, is an ensemble method that aims to decrease the variance compared to its components (Rocca, J., 2019) and avoid over fitting. It provides each classifier with a random subset of the original data set and aggregates their predictions, either by averaging or voting (Pedregosa, F. Et al, 2011). Bagging was used here to combine logistic regression classifiers to increase the accuracy of the hyper-tuned logistic regression base classifier.

Random Forest:

Random forest works by combining multiple decision trees into one classifier, increasing accuracy by reducing the effects a single decision tree has on the overall outcome. This means if a decision tree has low accuracy it will not have a significant impact on the overall accuracy of the random forest. Random forest utilises bagging which is what allows this method to achieve high classification accuracy. One of the differences between random forest and bagging is that random forest provides each decision tree classifier with a random selection of features from the data set. This differentiates each tree, therefore creating a greater ensemble which should, in turn, result in greater accuracy than bagging (Lutins, E., 2017). Studies have shown that in binary problems, such as predicting when a developer will earn low-income or high-income, feature selection that was based on the Gini index improved the performance of the models (Sarica, A., Cerasa, A. and Quattrone, A., 2017).

Ada boost:
Ada boost, short for adaptive boosting, works by building sequential models that aim to correct the errors from the previous model (Brownlee, J., 2020). One of the goals of boosting is to create models that are less biased than the individual classifiers that the ensemble method is composed of (Rocca, J., 2019). One study shows that AdaBoost has higher accuracy than KNN, Naïve Bayes, ANN and SVM classifiers (Zerrouki, N., Harrou, F., Sun, Y. and Houacine, A., 2018).

Voting Classifier:
Using a voting classifier allows for the combination of multiple classifiers regardless of their type. I.e a logistic regression classifier can be combined with a decision tree classifier to increase the accuracy compared to using either classifier on its own. Studies have shown that using voting improves the prediction accuracy compared to using single classifiers (Paris, I.H.M., Affendey, L.S. and Mustapha, N., 2010). Two majority voting classifiers will be used for this task, one which combines all base classifiers, and another which combines only the top three base classifiers. One downside of majority voting is its long run-time compared to running any of the classifiers individually.

Evaluation of Machine Learning Models:

Multiple performance metrics can be used to review the performance of models, this includes precision, recall, f1 score and accuracy. Precision is calculated as the number of true positives divided by the sum of false positives and true positives. Precision is best used when trying to lower the number of false positives. As an example, marking important emails as spam when they are not is more of a problem than missing spam emails as you don't want people missing important emails.

Recall is defined as the number of true positives divided by the sum of true positives and false negatives. Recall is the optimal performance metric when false negatives can be detrimental to the business. As an example, for fraud detection, false positives aren't overly problematic however false negatives can have serious consequences where potential fraud is missed by the system.

F1 score is calculated as $F1 = 2 * \frac{precison*recall}{precison+recall}$. It is defined as the harmonic mean of a model's precision and recall. F1 score is used when the cost of false negatives and positives both have tangible business costs. In this case, minimising false readings is a priority.

Accuracy is the ratio of correct predictions. It is also the most appropriate performance metric for predicting a developer's income as the overall accuracy is important for this task and false negatives or false positives have no specific detrimental effect on predicting a developer's income.

Accuracy of non hyper-tuned base models:

Table 7: Non hyper-tuned model performance.

Classifier	Accuracy (5 d.p)	Accuracy (%)	Precision (5 d.p)	Recall (5 d.p)	F1 score (5 d.p)
Decision Tree	0.80168	80.2	0.74611	0.79161	0.80072
K-nearest neighbour	0.81210	81.2	0.75039	0.84011	0.81820
Support vector machine	0.87440	87.4	0.82781	0.87876	0.87566
Logistic regression	0.87872	87.9	0.83208	0.88684	0.88039
ANN	0.85291	85.3	0.80326	0.85148	0.85353

As Table 7 shows, the accuracy of the base models left a lot to be desired. Model accuracy ranged from 80.2%, achieved by the decision tree, to 87.9%, achieved by the logistic regression classifier. While the support vector machine performed considerably better than the ANN however performed worse in all four performance metrics when compared to the logistic regression classifier. The ANN had 2.6% less accuracy than the logistic regression classifier however had 5.1% and 4.1% more accuracy than the decision tree and KNN classifiers respectively.

Accuracy of hyper-tuned base models:

Table 8: Hyper-tuned model performance.

Classifier	Accuracy (5 d.p)	Accuracy (%)	Accuracy increase from tuning (%)	Precision (5 d.p)	Recall (5 d.p)	F1 score (5 d.p)
Decision Tree	0.83422	83.4	3.2	0.78780	0.80576	0.83030
K-nearest neighbour	0.82507	82.5	1.3	0.75901	0.88330	0.83560
Support vector machine						
Logistic regression	0.87872	87.9	0.0	0.83208	0.88684	0.88039
ANN	0.88012	88.0	2.7	0.83349	0.88937	0.88190

Despite the ANN classifier having a long run-time and thus being harder to hyper-tune, it saw a significant increase in performance after hyper-tuning, increasing its accuracy from 85.3% to 88.0%. The ANN classifier was now

outperforming logistic regression in all four of the performance metrics tested. This was because logistic regressions performance did not improve after hyper-parameter tuning (Table 8).
All classifiers benefited from hyper-parameter tuning however some classifiers saw much more substantial increases in performance than others. The decision tree and ANN classifiers saw an increase in accuracy by 3.2% and 2.7% respectively. The KNN classifier has an accuracy increase of 1.3% and logistic regression saw no increase in percentage accuracy after hyper-parameter tuning. To improve the performance of the models further it would appear that further outlier detection would be needed.

All classifiers achieved an accuracy greater than 80% which shows that all models could be considered successful. However, the KNN classifier has the lowest performance overall with 82.5% accuracy, a 1.3% increase after being hyper-tuned. In all cases, the precision of the classifier was lower than the classifiers recall. This means that false negatives were less common than false positives which were to be expected given how the country can be a clear indicator of someone having a low income in some cases however cannot as easily be used to determine when a respondent has a high income. To explore this further, the following confusion matrices were created to analyse the frequency of false negatives and false positives.

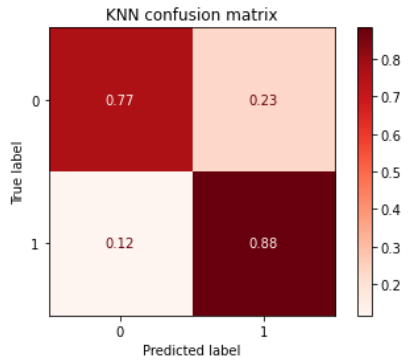


Figure 38: KNN confusion matrix.

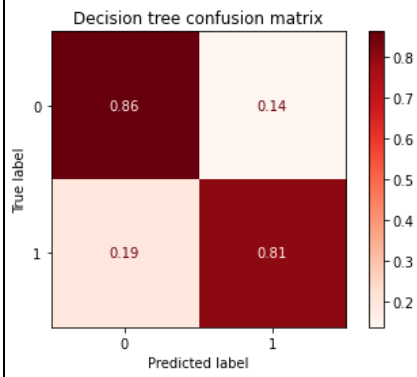


Figure 39: Decision tree confusion matrix.

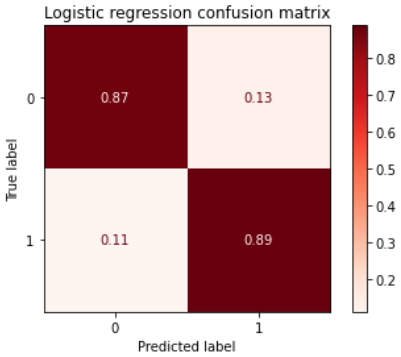


Figure 40: Logistic regression confusion matrix.

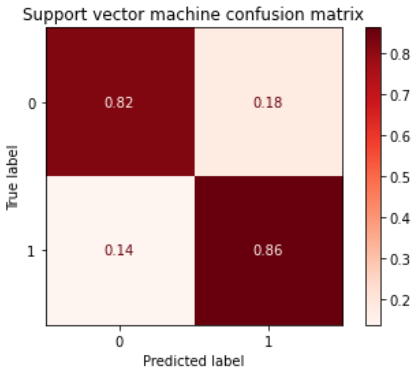


Figure 41: Support vector machine confusion matrix.

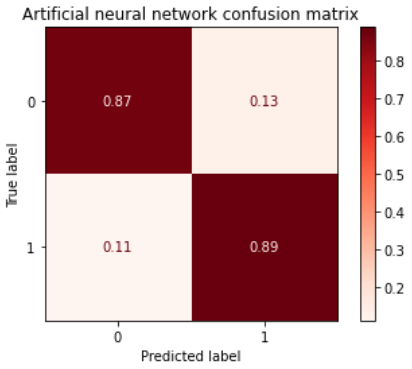


Figure 42: Artificial neural network confusion matrix.

The confusion matrices for the hyper-tuned models show how the model's performance varied considerably. KNN was the worst classifier for predicting low-income respondents, being incorrect 23% of the time. The decision tree was the worst at predicting high-income respondents, incorrectly labelling them as low-income 19% of the time. The logistic regression and artificial neural network classifiers both produced the same confusion matrix, showing how they achieved a good overall performance being 87% accurate when predicting true negatives and 89% accurate when predicting true positives. The support vector machine also performed extremely well however was 5% less accurate at predicting low-income developers and 3% less accurate at predicting high-income developers than both logistic regression and the artificial neural network.

The decision tree classifier was the only classifier that performed better at predicting low-income developers than high-income developers. There was a 10% difference between the worst and best performing classifier for predicting true negatives, KNN and ANN respectively. This shows how the KNN classifier significantly under performed in terms of accurately predicting true negative values however the KNN classifier was the third-best classifier at predicting high-income developers.

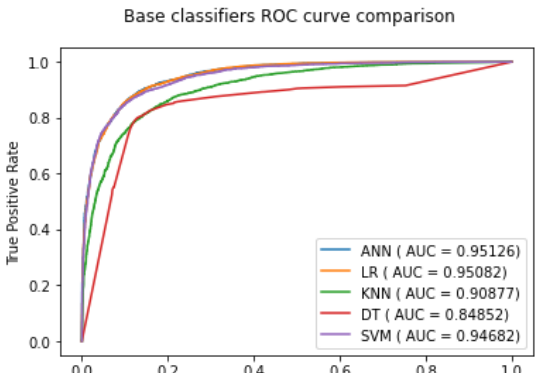


Figure 43: Base classifiers ROC curve comparison.

Figure 43 shows the ROC curves for all the base classifiers that were implemented along with their Area Under the Curve (AUC) values. AUC values range from 0, being the worse possible value and 1, the perfect value. Unsurprisingly, the AUC values for each of the hyper-tuned classifiers are mostly in the same order as the corresponding accuracy values for each classifier, however, the ANN has a higher AUC score than the logistic regression classifier. The decision tree had the lowest AUC score at 0.849 and the ANN had the highest at 0.951. Hence there is a large difference between the worst and best hyper-tuned classifier of 0.102.

Except for the decision tree, all hyper-tuned classifiers had an AUC score that was greater than 0.9 which shows that all the classifiers have high performance except for the decision tree.

Deep MLP evaluation:

The implemented deep MLP cannot be evaluated in the same way as the other classifiers. This is because the performance of deep learning can vary on re-runs and does not provide confusion matrices or ROC curves. The evaluation for deep MLP's does not show the precision, recall or F1 score of the model, however, does show accuracy and a new performance metric in the form of loss.

Table 9: Deep learning performance.

Test case	Loss	Accuracy
1	0.313935	0.873506
2	0.310289	0.878464
3	0.311056	0.875413
Average	0.311760	0.875794

As results from deep learning can vary when the model is re-run, the performance of the deep MLP was evaluated three times and an average value for loss and accuracy was produced. Table 9 shows the performance of the deep learning method that was used on each of the three runs as well as the average performance. The deep MLP performed exceptionally well however performed slightly worse than the hyper-tuned ANN.

Accuracy of ensemble models:

Table 10: Ensemble performance

Classifier	Accuracy (5 d.p)	Accuracy (%)	Precision (5 d.p)	Recall (5 d.p)	F1 score (5 d.p)
Random forest	0.84999	85.0	0.79853	0.85350	0.85135
Bagging (logistic regression)	0.87796	87.8	0.83107	0.88634	0.87967
Ada boost	0.87211	87.2	0.82613	0.87244	0.87288
Voting classifier (All base classifiers)	0.87961	88.0	0.83365	0.88608	0.88108
Voting classifier (SVM, ANN, LR)	0.87948	87.9	0.83324	0.88684	0.88105

Table 10 shows that the random forest classifier had the lowest accuracy out of all the ensemble classifiers at 85.0%. This is an increase of 1.6% over the hyper-tuned decision tree classifier, which had an accuracy of 83.4%. AdaBoost performed better, having an accuracy of 87.2% however bagging outperformed AdaBoost in all four performance metrics used and had an accuracy of 87.8%. Both majority voting classifiers performed exceptionally well, however, combining all hyper-tuned classifiers proved to increase performance more than combining only the top three hyper-tuned classifiers. The best performing ensemble method was the majority voting classifier that utilised all hyper-tuned classifiers, having an accuracy of 88%. Surprisingly, none of the ensemble methods used achieved a performance greater than that of the hyper-tuned artificial neural network, which also had an accuracy of 88%.

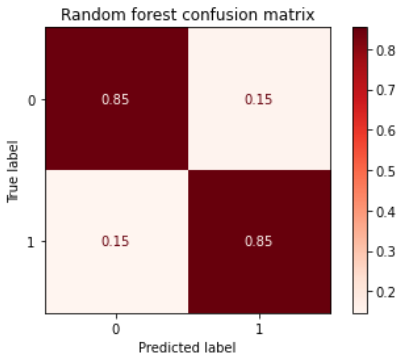


Figure 44: Random forest confusion matrix.

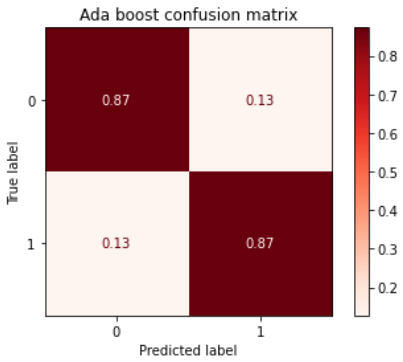


Figure 45: Ada boost confusion matrix.

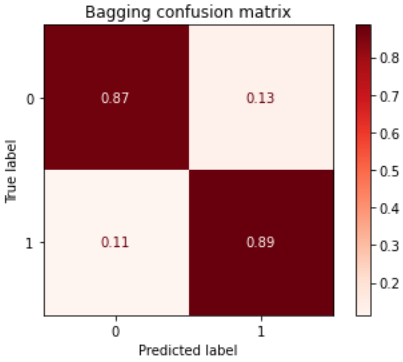


Figure 46: Bagging confusion matrix.

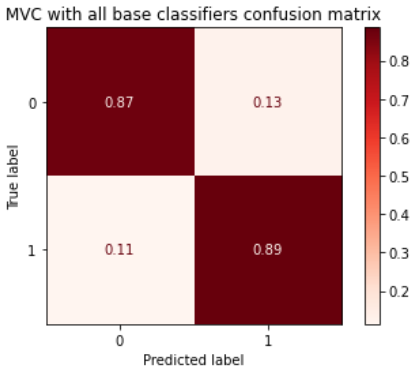


Figure 47: MVC with all base classifier's confusion matrix.

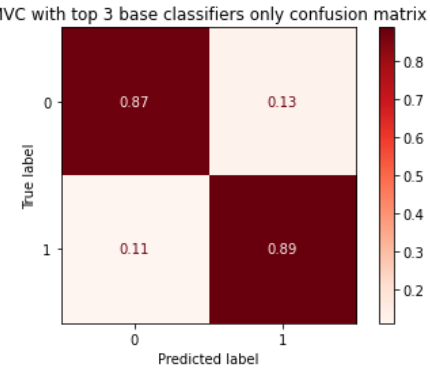


Figure 48: MVC with top 3 base classifiers only confusion matrix.

Figure 44 shows how the random forest was the worst-performing ensemble classifier at predicting true negatives, predicting that low-income developers were high-income developers 15% of the time. Random forest also had the lowest true positive percentage at 85%. Despite being the lowest true positive rate out of all the ensemble methods used, this was still a considerable increase over the hyper-tuned decision tree which has a true positive rate of 81%. This shows how the implemented random forest classifier could be considered successful despite performing worse than the other ensemble methods.

The confusion matrices provided further insight into what made the bagging classifier (Figure 46) slightly more accurate than the AdaBoost classifier (Figure 45), bagging had a true positive rate of 89%, a 2% increase over AdaBoost but both had a true negative rate of 87%. This also explains why the bagging classifier had a higher precision than AdaBoost.

The MVC that combined all classifiers () produced the same confusion matrix as the MVC that combined only the top three classifiers. This shows how the difference in performance between the two classifiers was minimal. They were also the same as the bagging classifier's confusion matrix.

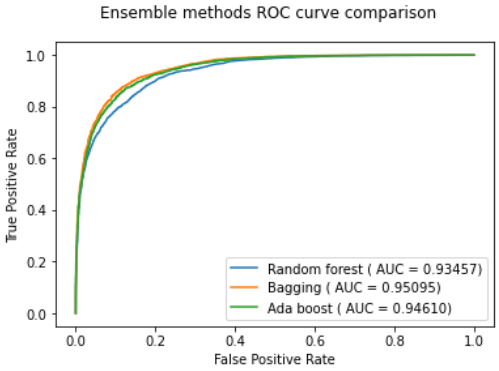


Figure 49: Ensemble methods ROC curve comparison.

Figure 49 shows the ROC curves for the random forest, bagging, and Ada boost ensemble classifiers. The two majority voting classifiers are not shown here as hard voting classifiers cannot produce ROC curves.

Bagging and boosting produced similar curves however, AdaBoost had an AUC score of 0.946 when rounded to three decimal places whereas bagging had an AUC score of 0.951. The random forest classifier had the lowest AUC score at 0.935. This shows how random forest performed worse than the other implemented ensemble methods in all performance metrics that were evaluated.

Overall, evaluation of the ensemble methods showed that random forest was the worst-performing ensemble classifier, followed by AdaBoost and bagging. The greatest performing ensemble classifiers were the majority voting classifiers, although less evaluation was performed on these classifiers as ROC curves could not be produced.

Discussions and Conclusions:

To conclude, my task was to apply the CRISP-DM methodology for machine learning to the stack overflow annual developer survey data set. I needed to analyse the data and remove outliers to be able to accurately predict whether a developer had low-income or high-income. I also intended to gain insight into the relationship between attributes and a respondent's salary using exploratory data analysis (EDA).

The analysis of developer salaries shows that there is a relationship between the country, education level, years of professional coding experience, developer type and salary. This has shown how salaries increase based on an individual's level of education and has allowed me to analyse the impact a master's degree or doctoral degree has on a developer's salary, which has encouraged me to seek further qualifications upon completion of my bachelor's degree. EDA also allowed for the economies of many countries to be visualised. Particularly India, which saw considerably lower salaries than the other frequent countries in the stack overflow developer survey, such as the United States and the United Kingdom. Furthermore, the analysis showed how the salaries for both men and women increased at a similar rate based on age and showed how generally, professional experience is worth more than nonprofessional experience.

The primary objective of this module has been to create machine learning models that can be used to accurately predict when an individual will have low-income or high-income. In this regard, I would consider the implemented models to be a resounding success. All hyper-tuned and ensemble classifiers achieved an accuracy that was greater than 80%, and in the best-case scenario, the majority voting classifier that combined the ANN, SVM and logistic regression classifiers achieved a maximum accuracy of 86.4%. Hyper-parameter tuning proved to be effective yet time-consuming, the long run-time primarily stemmed from using grid searches so one potential improvement to my models would be to use alternative hyper-parameter tuning methods that decrease run-time without decreasing accuracy.

This module has allowed me to develop insight into a broad range of areas. I have developed a deep understanding of the CRISP-DM methodology and the machine learning process and uncovered a passion for machine learning. This module has helped me understand a variety of machine learning models including decision trees, logistic regression, and artificial neural networks. As well as this, I have felt encouraged to explore machine learning further and expand my knowledge beyond the taught content of the module, leading to my use of support vector machines. Furthermore, I've understood the factors that affect an individual's salary which could prove to be invaluable in the future when I need to decide which area of computer science I would like to specialise in. It has also allowed me to see how salary increase with age and years of professional coding experience which has helped me visualise career growth in the industry. Gaining an understanding of economic poverty has also been an unforeseen benefit of this module, seeing how salaries differed between countries has been extremely insightful and has encouraged me to perform independent research on the conditions of different countries such as, but not limited to, India.

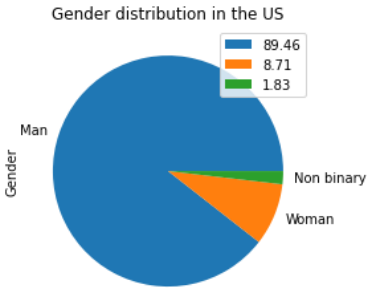
There would undoubtedly be room to improve the accuracy of the implemented models in the future. Hyper-parameter tuning proved to be difficult for the support vector machine, and the K-nearest neighbour classifier had considerably lower accuracy than the other classifiers so it is likely that both models could be improved. Furthermore, despite my extensive efforts to remove outliers, it would be fair to say that they do still exist in some form in my dataset, given more time I would like to analyse my data further as much of the outlier removal has been implemented using a series of mathematical equations that aimed to detect outliers based on a respondents age, years of professional coding experience, workweek hours and country. Implementing some form of outlier detection based on a respondent's education level or developer type would be the next logical step to remove outliers and it is possible that by doing this, greater accuracy could be achieved.

References:

1) Alade, Tola., 2018. *How to determine the optimal number of clusters for k-means clustering*, Cambridge Spark, viewed 18/04/2021, https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f.
2) Biceková, A. and Pusztová, Ľ., 2019. Data Mining from the Banking Sector´s Data. *Acta Informatica Pragensia*, 8(1), pp.18–37.
3) Bose, I. and Mahapatra, R.K., 2001. Business data mining—a machine learning perspective. *Information & management*, *39*(3), pp.211-225.
4) Brownlee, J., 2020. *Boosting and AdaBoost for Machine Learning*, Machine Learning Mastery, viewed 10/04/2021, https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/
5) Brownlee, J., 2020. *Recursive Feature Elimination (RFE) for Feature Selection in Python*, Machine Learning Mastery, viewed 18/04/2021, https://machinelearningmastery.com/rfe-feature-selection-in-python/.
6) Brownlee, J., 2016. *Logistic Regression for Machine Learning,* Machine Learning Mastery, viewed 08/04/2021, https://machinelearningmastery.com/logistic-regression-for-machine-learning/#:~:text=Gaussian%20Distribution%3A%20Logistic%20regression%20is,in%20a%20more%20accurate%20model.
7) Contreras, Yudith et al., 2018. Digital processing of medical images: application in synthetic cardiac datasets using the CRISP_DM methodology. *Revista Latinoamericana de Hipertension*, 13(4), pp.310–315.
8) Cunningham, P. and Delany, S.J., 2020. k-Nearest Neighbour Classifiers--. *arXiv preprint arXiv:2004.04523*.
9) Feng, J., Xu, H., Mannor, S. and Yan, S., 2014. Robust logistic regression and classification. *Advances in neural information processing systems*, *27*, pp.253-261.
10) Gajawada, S., 2019. *Chi-Square Test for Feature Selection in Machine learning,* towards data science, viewed 18/04/2021, https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223.
11) George, Seif., 2018. *A Guide to Decision Trees for Machine Learning and Data Science*, towards data science, viewed 08/04/2021, https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956
12) Great Learning Team, 2020. Why using CRISP-DM will make you a better Data Scientist? Viewed 22 Jan 2021. Available at <https://www.mygreatlearning.com/blog/why-using-crisp-dm-will-make-you-a-better-data-scientist>
13) Gupta, Prashant., 2017. *Decision Trees in Machine Learning,* towards data science, viewed 08/04/2021, https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.
14) Lutins, E., 2017. *Ensemble Methods in Machine Learning: What are They and Why Use Them?*, towards data science, viewed 10/04/2021, https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f.
15) Paris, I.H.M., Affendey, L.S. and Mustapha, N., 2010. Improving academic performance prediction using voting technique in data mining. *World Academy of Science, Engineering and Technology*, *62*, pp.820-823.
16) Pedregosa, F. Et al, 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, pp.2825-2830.
17) Ragab, D.A., Sharkas, M., Marshall, S. and Ren, J., 2019. Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, *7*, p.e6201.
18) Ribeiro, R. Et al, 2020. Predicting the tear strength of woven fabrics via automated machine learning: an application of the CRISP-DM methodology.
19) Rocca, J., 2019. *Ensemble methods: bagging, boosting and stacking*, towards data science, viewed 10/04/2021, https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205.
20) Rungta, K., 2020. *Back Propagation Neural Network: Explained With Simple Example,* Guru99, viewed 07/04/2021, https://www.guru99.com/backpropogation-neural-network.html#:~:text=Back%2Dpropagation%20is%20the%20essence,reliable%20by%20increasing%20its%20generalization.
21) Sagi, O. and Rokach, L., 2018. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(4), p.e1249.
22) Sarica, A., Cerasa, A. and Quattrone, A., 2017. Random forest algorithm for the classification of neuroimaging data in Alzheimer's disease: a systematic review. *Frontiers in aging neuroscience*, *9*, p.329.
23) Shafique, U. and Qaiser, H., 2014. A comparative study of data mining process models (KDD, CRISP-DM and SEMMA). *International Journal of Innovation and Scientific Research*, *12*(1), pp.217-222.

24) Shouman, M., Turner, T. and Stocker, R., 2012. Applying k-nearest neighbour in diagnosing heart disease patients. *International Journal of Information and Education Technology*, *2*(3), pp.220-223.
25) Wang, M., Cui, Y., Wang, X., Xiao, S. and Jiang, J., 2017. Machine learning for networking: Workflow, advances and opportunities. *Ieee Network*, *32*(2), pp.92-99.
26) Wirth, R. and Hipp, J., 2000, April. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (pp. 29-39). London, UK: Springer-Verlag.
27) Zerrouki, N., Harrou, F., Sun, Y. and Houacine, A., 2018. Vision-based human action classification using adaptive boosting algorithm. *IEEE Sensors Journal*, *18*(12), pp.5115-5121.
28) Mithmrakumar, M., 2019. *How to tune a Decision Tree*, towards data science, viewed 21/04/2021, https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680.
29) Tiwari, S., 2020. *Activation functions in Neural Networks*. GeeksforGeeks, viewed 21/04/2021, https://www.geeksforgeeks.org/activation-functions-neural-networks/.

Appendix



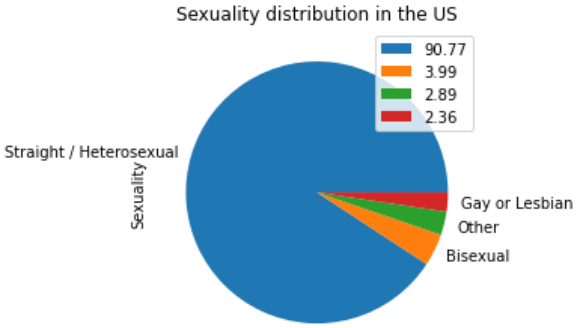


Table 11: Countries with economies high enough for low salary anomoly detection to be used.

United States
United Kingdom
Germany
Canada
France
Netherlands
Australia
Sweden
Switzerland
Norway
Poland
Spain
Italy
Austria
Israel
Belgium
Ireland
Denmark
Finland
New Zealand
Japan
Iceland
Luxembourg
Portugal
Lithuania