COMP20121

FOUNDATIONS OF AI & MACHINE LEARNING

Contents

Introduction

Initial Objectives

Within this report, the "2024 Stack Overflow Annual Developer Survey" data set will be used for predicting high-income developers depending upon their provided information. The information of the respondents includes characteristics such as age, work experience, and organisation size. The relationship between these characteristics and the developer's income will be analysed, with select characteristics being used to train aconsidered as someone who's income is greater than the median value of "ConvertedCompYearly" – the yearly pay of the respondent converted from their own currency. The expected insights to gain from this report includes determining if there are any data patterns contained within the Stack Overflow Annual Developer Survey, and how such data or characteristics can be grouped and used to train a binary classification model.

The CRISP-DM Methodology and its Applications

Following an exact methodology is not a common practise within the field of Data Science. From Salz's survey, 82% of scientists were not employing an explicit process. Despite that, it should be noted that 85% of the survey respondents would support the usage of a more well-defined methodology as being beneficial (Saltz, et al. 2018). Further support for a well-defined methodology derives from its success in large-data projects (Schröer, Kruse and Gómez 2021). Additionally, a well formulated sequence of steps would be particularly beneficial in ensuring an efficient and effective process (Wirth and Hipp 2000). Hence, employing a methodology appears suitable for this project.

As for which methodology to employ, CRISP-DM will be the choice for this project. CRISP-DM is the Cross-Industry Standard Process for Data Mining consisting of 6 key phases. The flow of these phases within the model can be observed within *Figure 1*.

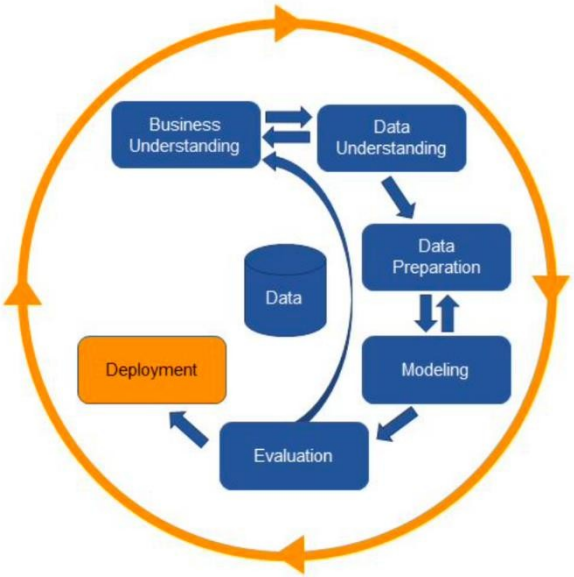

Figure 1: *The CRISP-DM Flow Cycle.*

The **Data Understanding** phase - once the initial business value has been assessed, the assessor should proceed to gain an understanding of what the data contains. This will involve visualisation of the data via graphs or charts to identify characteristics or patterns, from which early observations can be made of interesting subsets or patterns. Alongside this, an evaluation of the data quality should be performed – making note of its errors, flaws, or other values that may require cleaning (Huber, et al. 2019, Schröer, Kruse and Gómez 2021). This will be performed upon the provided data in this report to provide early insight into interesting characteristics that can be followed up on in the next stage.

Moving into the **Data Preparation** phase, the prior observations should make it possible to evaluate the notable characteristics or data patterns. Inclusion criteria can be put forward to determine what data is to be utilised by the Machine Learning model. The flaws in the data must also be accounted for and cleaned – as they may otherwise skew trends or produce unexpected results in a Machine Learning model. Other data issues may arise from poor or non-standardised formatting, which must also be cleaned (Schröer, Kruse and Gómez 2021, Chapman, et al. 1999). With regards to this project, a strong set of criteria to select the

predictor data, alongside ensuring a high quality of data through thorough cleaning should enable a high-quality model to be produced.

The **Data Modelling** phase will then apply the machine learning models onto the procured data set. This will involve selecting a model technique, building and assessing its performance. The choice of model will be influenced by the data-mining goal and the actual data involved. Additionally, multiple models may be applied to the data and their performance re-assessed as tweaks are made to produce a more suitable final model (J. S. Saltz 2021, Schröer, Kruse and Gómez 2021).

Proceeding into the **Evaluation** phase, once a suitable model (or multiple) has been produced, the model's data mining results should then be evaluated with consideration to the original business goals. The applicability of models should be stated, alongside a review of if any business goals remain unsatisfied and further actions to be put forward as a potential solution. Should the evaluation of the models be favourable, then the **Deployment** phase handles using and applying the end-results of the process. This may be a report of the results from the data mining, or a software component that can be applied to a dataset. This phase may also extend into continued maintenance and monitoring of the model (J. S. Saltz 2021, Wirth and Hipp 2000, Schröer, Kruse and Gómez 2021).

Applying the CRISP-DM Methodology to my Project

Now that an understanding of the methodology has been established, the first stage can be considered and integrated into this report. This begins by re-assessing the goals of this project from a business perspective. As previously stated, the initial goal was to predict high-income developers using the Stack Overflow Annual Developer Survey. Where stack overflow may find benefits in being able to predict high-income developers is with aspects of the site dedicated to jobs and companies. More personalised and suitable suggestions could be produced by predicting developer income based upon some characteristics that may already be known about the user. Additionally, the process of exploratory data analysis and clustering can be expected to result in some patterns or common characteristics amongst Stack Overflow users. This can then be used to improve services or provide new services for large groups of users with shared characteristics that may not immediately be obvious.

The further sections of the methodology will be continuously applied throughout this report. Exploratory Data Analysis and Clustering will build up the understanding of the data contents. Those sections will also provide an opportunity to discuss the data transformation changes, and how redundant data has been reduced from the data set. The modelling phase will regard this reports implementation of 3 initial models, followed by a further 2 ensemble models. This will also include hyperparameter tweaking and evaluation of the models' accuracy as these values are altered. This will culminate in a final evaluation of each model independently and then comparatively, to determine the overall most successful model. The deployment phase will then make an evaluation of the final accuracy results and potential changes that may be beneficial to investigate in further reports.

Exploratory Data Analysis and Data Understanding

This report will be using the "2024 Stack Overflow Annual Developers Survey" – a spreadsheet of data collected from respondents to a survey hosted by Stack Overflow (Stack Overflow 2024). For this survey, there would be 65,447 responses from 185 different countries, with the survey being designed to acquire a vast range of experiences. Since the survey was open to all Stack Overflow users, this means it may also include people who are not actively working or working as a developer of some form. As a result, these records will have to be removed, as the data will not be usable for this model – given they will not have any reported annual income to predict. With the remaining suitable respondents, there were 87 questions in the survey, with notable categories to observe being 'Education, Work and Career', 'Technology and Tech Culture', 'Stack Overflow Usage + Community', 'Artificial Intelligence', and finally 'Professional Developer Series (Optional)'.

Prior to any initial analysis, it is important to note the contents and characteristics of the constituent data and ensure that it is understandable from a quick observation. Fortunately, many of the data headings are obvious, and a schematic is provided for translating between question and database headings where required. A common format is applied for the data headings of multiple-option questions such as "Which embedded systems have you worked with" (ID340). The results are split into the headings named using the format [QName][Option]. Despite this, only some of those data columns will be required – primarily the ones specifying tools or technologies the respondents have worked with over the last year. As a result, they will require changing. Additionally, there are issues from data headings belonging to the "Professional Developer Series (Optional)" section, with there being 9 individual headings named as "Knowledge_[index]". Fortunately, this set of questions are unlikely to be influential factors, and as a result, providing new headings is not an immediate aim, as the data can instead be dropped entirely. With regards to the remainder of the columns, some changes will be required to provide them more meaningful or obvious headings. For example, "AISelect" refers to the AI tools currently being used by the respondent in their development process. A sample of the updated headers is provided in Figure 2.

EmbeddedHaveWorkedWith	RecentEmbeddedSystems
MiscTechHaveWorkedWith	RecentOtherFrameworks
ToolsTechHaveWorkedWith	RecentDevelopmentTools
AISearchDevHaveWorkedWith	RecentAITools
SOVisitFreq	FrequencyOfVisitingSO
SOPartFreq	FrequencyOfParticipatingSO
SOHow	HowDoYouUseSO
AISelect	AITools
AISent	StanceOnAITools
AIThreat	DoesAIThreatenYourJob
ICorPM	IndividualOrManager
CompanyTech	ProfessionalTech

Figure 2: Sample of the updated data header names

All data analysis must be done within the perspective of the goals of the project. In this case, an analysis of what characteristics can predict a high-income developer. A quick analysis of the Converted Annual Compensation data provides us with a median income of 65,000, resulting in 11707 respondents being classed as high-income developers. However, the data contains numerous outlier values that will otherwise heavily skew models or graphs. The highest value within the annual compensation data is over 16million, which can be attributed to an error in currency conversion. However, this is not the only value that could skew the results, and both excessively high and low values will need to be checked. Whilst the final observations will be a binary check of >Median or <=Median, it is still critical to ensure the removal of NaN values, alongside removing any excessive skew being applied to median or quartile values, to produce an effective model.

Data Quality and Characteristics

Most of the data is categorical, however there are discrete values contained within the results, such as with the 'Years of Work Experience' and 'Years Coding' responses. This categorical data may need to be reformatted into numerical data to be more readily applied to a machine learning model. Fortunately, this is unlikely to encounter many issues. The survey primarily provides the respondents with listed options to select from, which can easily be converted into numerical values. For example, a dictionary can be created for programming language names to replace them with numeric values.

```
{'Bash/Shell (all shells)': 1, 'Go': 2, 'HTML/CSS': 3, 'Java': 4,
 'JavaScript': 5, 'Python': 6, 'TypeScript': 7, 'Kotlin': 8, 'C#':
 9, 'C': 10, 'C++': 11, 'PHP': 12, 'PowerShell': 13, 'SQL': 14,
 'Lua': 15, 'Rust': 16, 'Swift': 17, 'R': 18, 'Ruby': 19,
 'Crystal': 20, 'Delphi': 21, 'VBA': 22, 'Visual Basic (.Net)': 23,
  'F#': 24, 'Clojure': 25, 'Julia': 26, 'Scala': 27, 'Zig': 28,
 'Perl': 29, 'MATLAB': 30, 'Dart': 31, 'OCaml': 32, 'Assembly': 33,
  'Solidity': 34, 'Elixir': 35, 'Cobol': 36, 'Fortran': 37, 'Lisp':
 38, 'Prolog': 39, 'Haskell': 40, 'Ada': 41, 'Erlang': 42,
 'Groovy': 43, 'nan': 44, 'GDScript': 45, 'MicroPython': 46,
 'Objective-C': 47, 'Nim': 48, 'Apex': 49, 'Zephyr': 50}
```

Figure 3: A Dictionary containing all Programming Lanuages mentioned in the Languages Question (ID233) responses

Most questions within the survey were optional, which indicates that there will be a common presence of unsuitable data to be cleaned from the dataset. And upon counting, there are **2,890,957** NA options. Shown to the right in *Figure* 2 are a selection of data fields and the percentage of their values left as NA. However, this value will be increased due to the formatting of the data set. As mentioned previously, multiple-option question results are split into multiple distinct data columns. Despite the inflated value, this still shows the clear distinction between required and optional questions from the survey and shows why the cleaning of the data fields will be necessary. Additionally, it could be utilised as an inclusion criterion – high rates of NA values may cause a lack of data, making the column difficult to apply to a machine learning model.

ResponseId	0.000000
MainBranch	0.000000
Age	0.000000
Employment	0.000000
RemoteWork	16.243678
...	...
JobSatPoints_11	54.994117
SurveyLength	14.141214
SurveyEase	14.055648
ConvertedCompYearly	64.177120
JobSat	55.481535

Figure 4: Percentage of values in ata Fields which were NA for a selection of data headings

Alongside high-NA rates, we can also assess and then reduce the data pool that needs to be examined and cleaned by removing unnecessary parts of it. Primarily, data which is derived from subjective or opinion-based questions from the survey. Since these questions are unlikely to have any bearing upon the income of the respondents, they can be safely removed from the data set without majorly impacting our model later.

Data Cleaning

A mentioned, there are initially **2,890,957** NA/NaN values within the data set which must be cleaned to produce usable data. Starting with the most crucial data field, the Converted Annual Compensation, the NaN values will be removed, alongside any outliers. This should see a significant reduction to the remaining NA/NaN values. This will be additionally beneficial as it will prevent the unnecessary cleaning of data which does not have an associated Annual Compensation value – making it unusable for this project.

Hence, for this report, values exceeding the top and bottom 2.5% quantiles will be excluded. This will refer to annual compensation values below 1118.65, or above 263,273.25. Figure 4 on the right demonstrates the results from cleaning the data of these values. The median has remained at 65,000, but the total number of high-income developers has been reduced to 11121 (a reduction by 5.006%). The 22263 remaining cleaned responses will now be utilised throughout the rest of the report. However, future cleaning may reduce that number further.

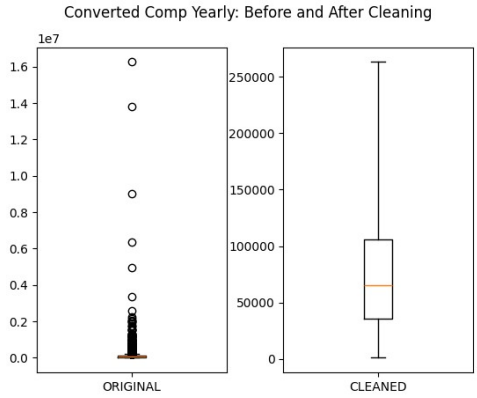

Figure 5: Before and After cleaning the Annual Compensation data

With regards to the remaining NA/NaN values, the total has been reduced to 608303 or 21.04% of the original quantity. Additionally, the mean percentage of NA values in all data columns is now only 8%. This is a significant improvement from the initial data set.

EdLevel	7.109570	EdLevel	0.000000
LearnCode	7.561844	LearnCode	0.047367
LearnCodeOnline	24.752853	LearnCodeOnline	5.922349
TechDoc	37.495989	TechDoc	9.746818
YearsCode	8.507647	YearsCode	0.062646
YearsCodePro	21.127019	YearsCodePro	0.119180
DevType	9.155500	DevType	0.047367
OrgSize	27.437468	OrgSize	0.035143
PurchaseInfluence	27.550537	PurchaseInfluence	0.055006
BuyNewTool	30.950235	BuyNewTool	1.298761
BuildvsBuy	33.735695	BuildvsBuy	2.117744

Figure 6: Before and After removing NaN and outlier Annual Compensation values

This can now be further improved by dropping the redundant fields from the remaining data. This will be the subjective opinion-based question data as previously mentioned, alongside duplicated and thus redundant data. This includes questions about the survey itself such as “SurveyLength” and “SurveyEase” (IDs 132 and 133 respectively). With regards to the duplicated data, fields such as “LanguagesAdmired” will contain the same information as “LanguagesHaveWorkedWith”. As a result, only one of these needs to be retained. The latter will be chosen as it involves all the languages the respondents have worked extensively with in the last year. The resulting changes can then be compiled into a new CSV Data set which can be used in place of the original for ease of Further management. The dataset can be reduced from 115 columns to just 42, permitting a further decrease to only 107557 NA values, equivalent to only 3.72% of the original count. Below in figure 7 is a breakdown of the remaining null values in each column – columns with no null values are not shown.

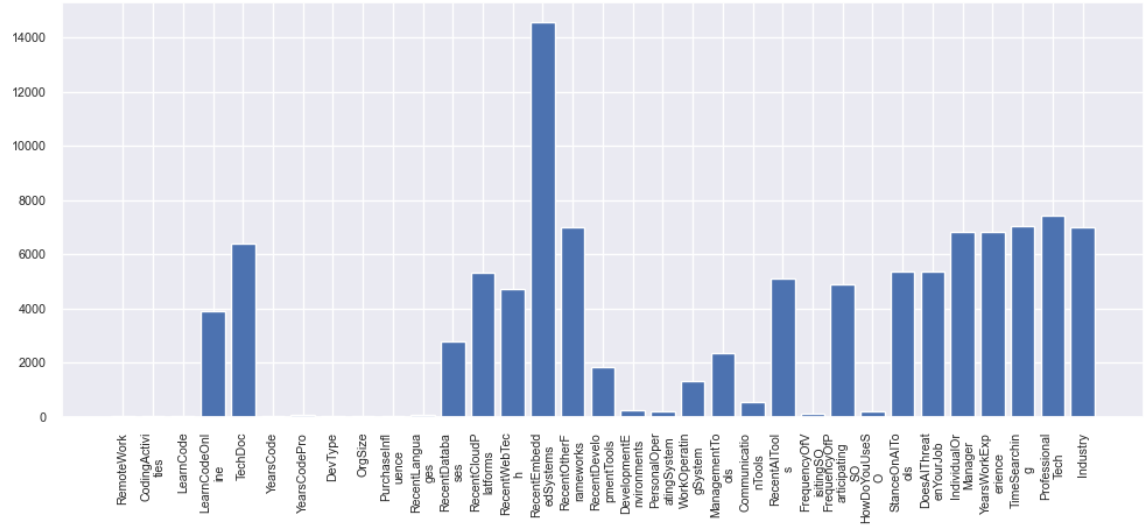

Figure 8: Null Value counts from select data fields

A secondary cleaning can then be done for Null values in data columns where the survey respondent did not select one of the available options. For suitable columns the Na value can be assigned to “None”, for example with the selection of Databases or Embedded Systems. These improvements can be seen in Figure 8. Finally, the remaining null values can be instead set to the modal value of their respective data fields.

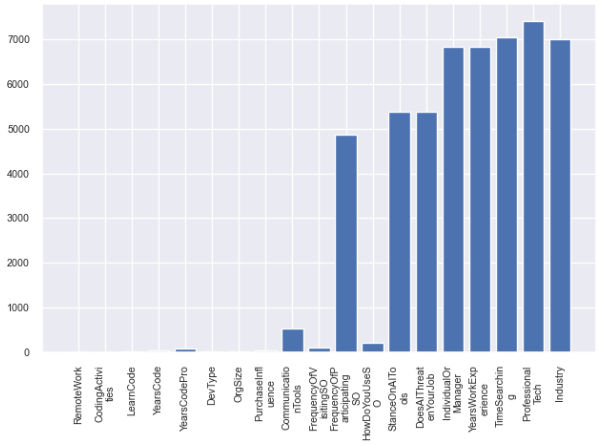

Figure 7: Remaining null value counts after replacing with None

With all null values now removed, there is one final aspect left to clean for – which is ensuring that all remaining respondents are developers by profession. Upon counting the data in “MainBranch”, we find that

1801 respondents marked themselves as not being primarily a developer. As the purpose of this report is to identify high-income developers, that results in these survey results being obsolete. Hence, they will also be dropped from the remaining database. The median does not change from this action, and remains at 65,000, with a remaining 20,462 valid responses.

Exploratory Data Analysis

In order to ease into the process of selecting the final model attributes, some initial exploratory analysis will be beneficial. As all data must be explored within the context of the report's goal – predicting developers earning above the median annual compensation – a review of this data is most important. As previously established, the median annual compensation is 65,000, resulting in 10324 survey respondents being classed as high-income developers, or 50.45% of the total respondents.

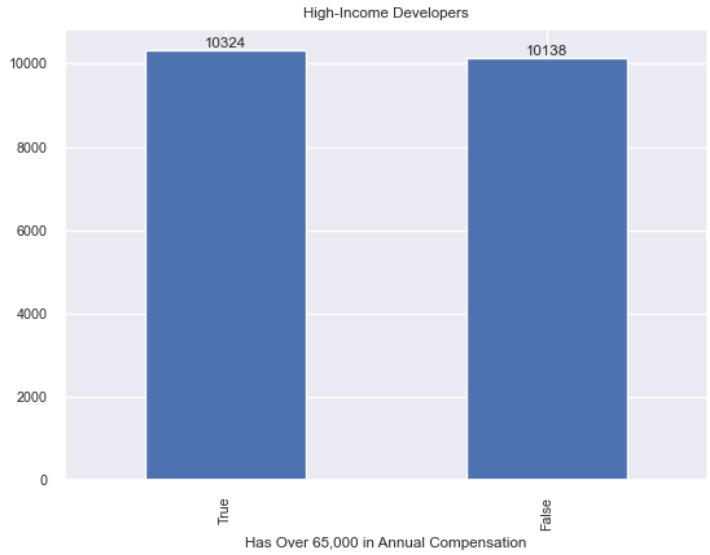

Figure 9: Graph of the High-Income Developer split within the respondents

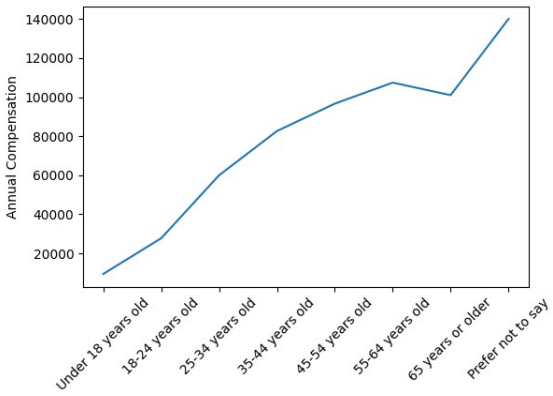

Figure 11: The Median Annual Income for each Age Range

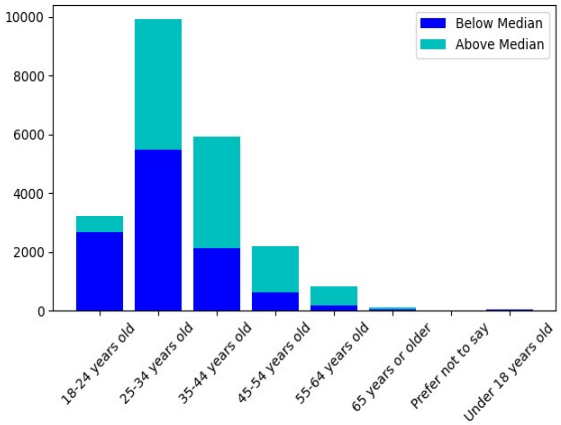

Figure 10: Counts of Above and Below Median income developers for each age group

Moving onto the first values to analyse, this will be the respondents' age. This was a non-optional categorical question, and this results in no data requiring cleaning or null values to remove. Thus, a comparison can be made quickly to observe the median pay for each specified age range. Looking to figure 5, the median pay is shown to increase as the respondents age increases. Notably, there is a slight dip as respondents reach the retirement age at 66, with the highest median value belonging to the "Prefer not to say" category. This is supported by an additional comparison with the counts of respondents above and below the median value for each group. The proportion of high-income developers evidently increases with age. This also gives further reason to the unexpectedly high medians for "Prefer not to say", as the sample size is significantly smaller, preventing a more natural distribution of wages lowering the medians. This could potentially have also affected the "65 years or older" group in the opposite manner – with an underrepresented median value.

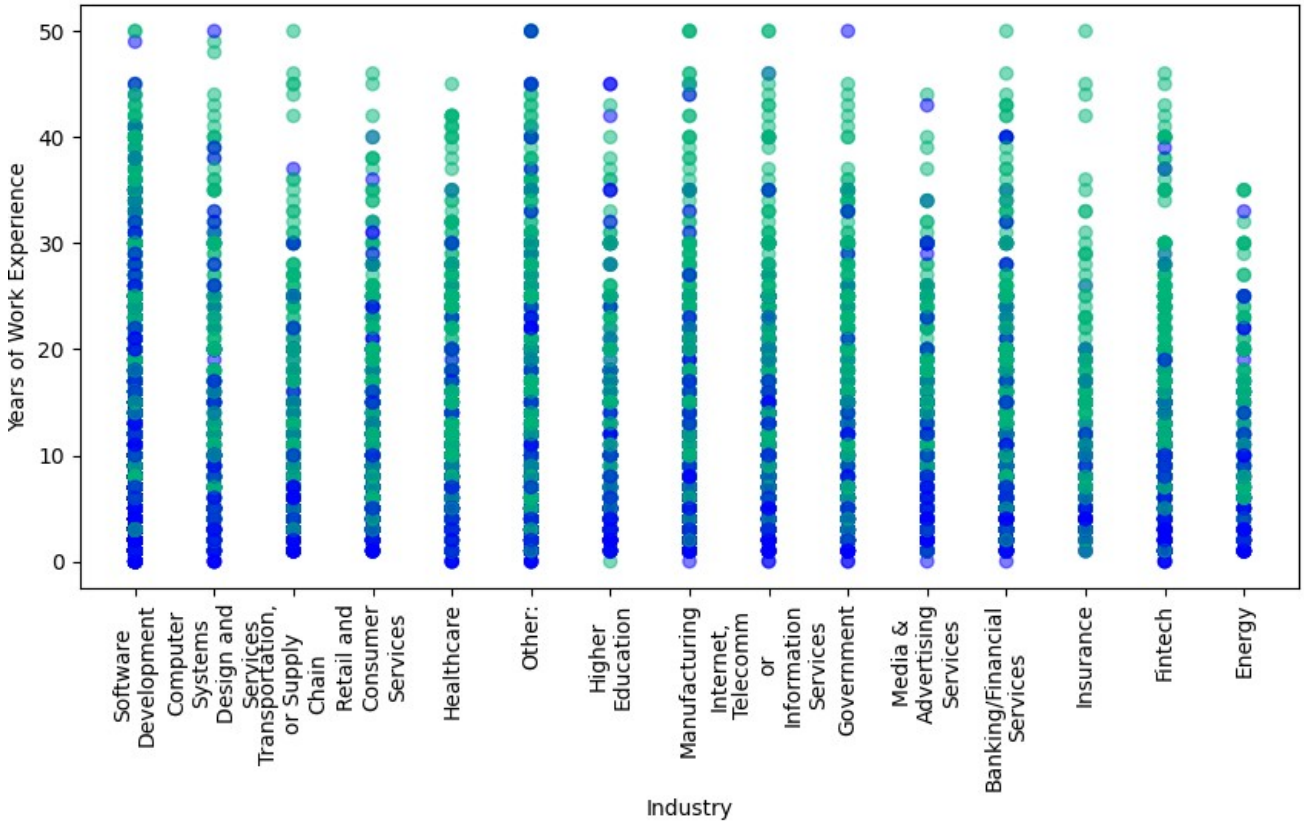

Figure 12: Years of Work Experience Against Industry, colour coded to show high-income developers (Green)

Further interesting patterns can be observed by using the respondent's industry and work experience information. Figure 12 was produced by plotting the two against each other and then applying a colour based upon the respondent's annual compensation being above / below the median. This allows for clear observations about how work and industry correlate to wages, with industries like Software Development generally requiring more years of experience to surpass the median wage, and others such as Fintech and Insurance requiring less. This comparison will be beneficial, as rather than a plain assumption of more experience correlating to higher wages, the machine learning model can also consider how the industry may influence more or less experience being required to earn above the median annual compensation. From a visual observation of the data, it can also be concluded that most respondents with 15 or more years of experience will earn above the median income. The graph does also present a few outlier values, such as the respondent with 0 years of work experience and yet above median pay in Higher Education. However, these values may have resulted from respondents mis-interpreting the original question, and answering with not their total years of experience, but years of experience for their current sector.

Selected Data Values

As previously analysed during the initial data analysis, three data columns that immediately stood out for analysis were the Age, Industry and Years of Professional Coding values. The other values that will be included are the respondent's developer type, organisation size, purchase influence, employment status and work location – these are expected to follow similar patterns to Industry, with particular roles or countries affecting income. And finally, the respondent's education level will also be used.

A comprehensive list of all retained data values will be stored within the appendix. This will also contain any further reasoning behind why values were chosen.

Cluster Analysis

The next step of evaluation and pre-modelling is to perform cluster analysis. Through clustering the data, we can produce groups and subsets of data that share relatively similar characteristics. The similarity of data and its clusters can be assessed by their plotted distribution and distance. However, cluster analysis will first require transforming data from Categorical values into numerical values, and then determining the most suitable parameters, alongside optimal cluster amounts.

Data Transformation and Normalisation

A lot of the remaining data is either Categorical, or list-based data. This will require correcting to more suitable apply models and clustering onto the data. The contents can also be assigned to either Nominal or Ordinal data, whereby nominal has no expected ordering, whereas ordinal does. It is in our interests to ensure that encoded nominal data does not end up misinterpreted by the machine learning model, by assuming that encoded value 1 is close to encoded value 2. This may be done by splitting unique data values into individual dummy columns. Furthermore, it will be beneficial to scale numerical data to prevent scales affecting models. 32 Years of Experience is a wholly different scale to an organisation size of 2999 people. But data scaling is also especially sensitive to outlier values, and so caution must be taken that those are cleaned beforehand. The dataset also contains a lot of listed values, meaning that for data transformation the lists must first be broken down before being able to be treated as nominal or ordinal values.

As a sample of the transformation methods involved, we will cover the data columns for “Age”, “Employment” and “Country”, as those contain a variety of challenges to deal with. Starting with Ages, those can easily be converted into an ordinal form for each group. A new column is then produced to contain the new values. However, a standard numerical label will not be applied. Instead, the midpoint age for each group will be used as the group’s numerical value – for example, the group 18-24 will adopt the encoded value of 21. This method can also then be applied to the Organisation Size. For the Employment field, due to it being composed of listed categories, specialized transformation of the data will be required. The unique values in the column must first be retrieved by splitting the listed data in the column. Then, the unique values can each be used as a column of binary true/false data in the modified dataset. The final example, “Country”, demonstrates how model-based transformations will be suitable. The column could be encoded using numerical labels; however, this would introduce a relationship between close numerical values – even if no such relationship is intended. This would negatively affect models such as K-Nearest Neighbour, but have no impact upon models such as a Decision Tree. But if converting each unique country into a new column, that will result in an excessive amount of columns. As a result, alternative encoding methods such as frequency encoding could be used – accepting the introduction of a relationship between the countries encoded values, but with regards to their frequency of response. Alternatively, use only a subset of the countries – likely the most frequent – as columns, and the remainder in an “Other” column. This is something that should be determined based upon the model.

	Age	AgeOrdinals	Employment	EmploymentOrdinals
0	18-24 years old	1	Employed, full-time;Studen...	3
1	35-44 years old	3	Employed, full-time	3
2	35-44 years old	3	Independent contractor, fr...	1
3	25-34 years old	2	Employed, full-time;Studen...	3
4	35-44 years old	3	Employed, full-time	3
...	...	...	...	...
20457	25-34 years old	2	Employed, full-time	3
20458	45-54 years old	4	Employed, full-time	3
20459	35-44 years old	3	Employed, full-time	3
20460	18-24 years old	1	Student, full-time;Employe...	2
20461	55-64 years old	5	Employed, full-time;Indepe...	3

Figure 13: Comparison of Categorical Ordinals and the new Numerical Values

K-Means Clustering

One means of clustering is K-Means clustering, whereby data points are grouped by determining their "distance" from a given central data point. In the standard K-Means algorithm, these initial centre points are chosen randomly from the data set. Once the data has been grouped to these cluster points, a new central point will be produced from the mean position of all data points within the cluster. This is repeated until the central points do not change – at least within some given tolerance level. However, the initial selection of random data points can cause poor clustering results. Poor clustering will then impact any observations being made upon the cluster characteristics. Hence, a modified algorithm, KMeans++, will be applied. After selecting the initial random cluster centres, it will then evaluate each data point in the random clusters. The further away it is from the current centre, the more likely it is to become the new centre, allowing for the cluster centres to be pushed away from each other for more suitable clustering.

Prior to clustering the data, it will be split into two initial groups. This split will be derived from the "ConvertedCompYearly", with the above-median income developers having their characteristics analysed separately from those who are below-median income. This is with the intention of providing better insights into the characteristics producing above-median income developers by isolating them from the others.

The other remaining critical point is to determine the optimal number of clusters. It is necessary to avoid over or under-splitting the data into various cluster groups for observation. To do this, the Elbow Method – a measure of distortion – can be used. The distortion is the relative sizes of each cluster, with a reduced distortion indicating a more evenly distributed split of values within the clusters. Hence, the objective of applying the Elbow Method is to determine the point at which the reduction in distortion slows from introducing more clusters. The results of this can be viewed in Figure 14, determining that 5 clusters is most suitable for grouping the characteristics of those earning below the median income, whilst 4 is better suited to grouping those with above-median income.

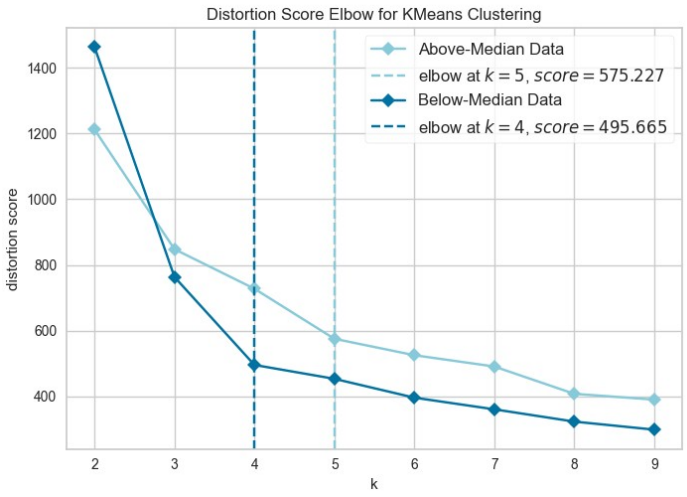

Figure 14:Determining the optimal cluster counts (k) for Above and Below-Median Income Data

Below are figures demonstrating the results of the data clustering for each of the two data groups. Each group has one graph to display each clusters' centre points, alongside the data of two of the cluster groups as a sample. Each cluster centre point is labelled numerically and shares the same colour as any plotted data. The clusters' data points use a base alpha value of 0.5 to better show the density of the clustering. This is adjacent to a secondary graph to provide more visible information upon the values of each cluster centre. This is done to provide an additional visual aid when comparing the characteristics of each cluster.

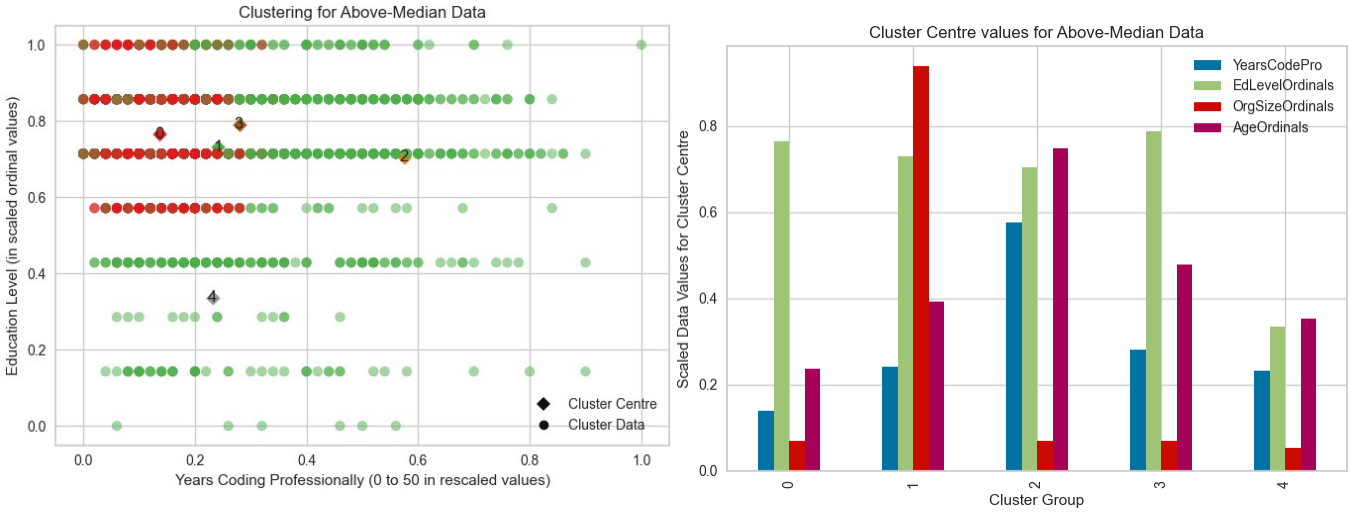

Figure 17: (Shown Left) The Data Clusters produced from the Above-Median data group. (Shown Right) The values used for each clusters' centre point

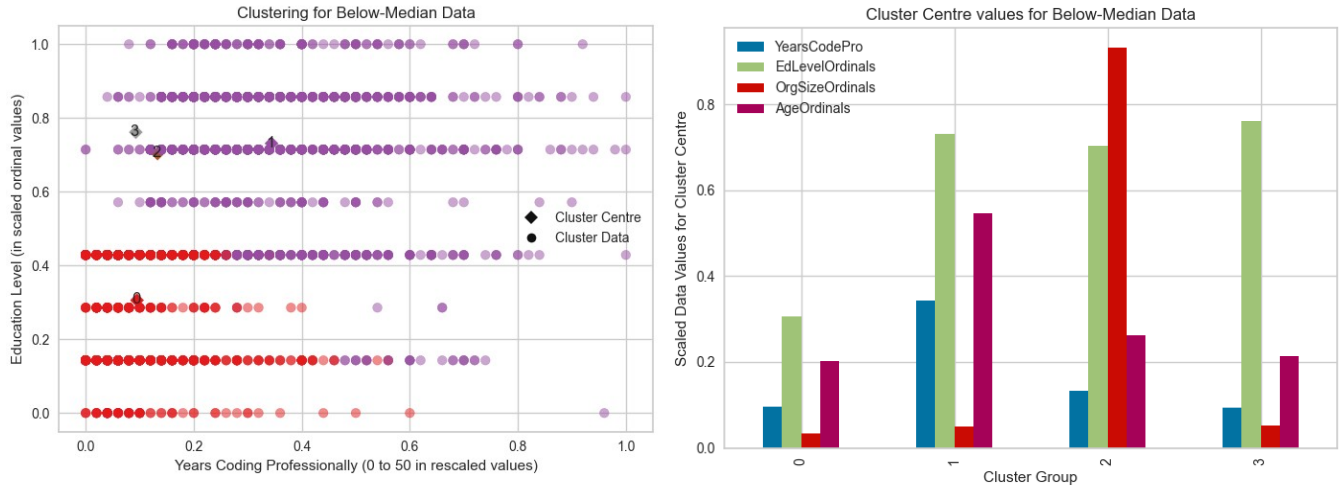

Figure 16: (Shown Left) The Data Clusters produced from the Below-Median data group. (Shown Right) The values used for each clusters' centre point.

	YearsCodePro	EdLevelOrdinals	OrgSizeOrdinals	AgeOrdinals
0	6.870126	5.354403	683.789937	28.342138
1	12.099182	5.116564	9392.772239	35.809816
2	28.804931	4.925308	680.729152	52.986947
3	14.040015	5.526121	683.518340	39.947017
4	11.639640	2.344144	517.677477	33.911712

	YearsCodePro	EdLevelOrdinals	OrgSizeOrdinals	AgeOrdinals
0	4.748584	2.139864	336.642695	26.691959
1	17.202026	5.117271	506.827825	43.267591
2	6.646429	4.924107	9323.525446	29.642857
3	4.635045	5.330357	519.796875	27.300595

Figure 15: Unscaled values for each cluster centre. Top belonging to Above-Median, and the groups below belonging to Below-Median

Cluster Observations and Evaluation

The most pronounced observations present themselves when comparing the typical characteristics of the cluster groups from Above-Median against the Below-Median data. Using “YearsCodePro” as the example, in the Below-Median data the central value of each group is around 0.1, translated to 5 years, with the average resulting in 8.3 years due to the outlier cluster group 1. Having 17.2 years as its central value, this marks cluster group 1 as a clear outlier with regards to the total years of professional coding. With regards to the “YearsCodePro” values of the Above-Median cluster centres, the average centre has 14.7 years of coding. and group 2 goes far beyond the typical observed centre values reaching 28.8 years. Additionally, the “YearsCodePro” values are consistently higher than the Below-Median groups - an indication that the “YearsCodePro” is a good characteristic for determining high-income developers.

Another observation that may be made initially is for the values of Organisation Size and Education Level. Graphically, the values do not indicate significant change, appearing largely identical across clusters, and between the above-median and below median data. But once reconverting to the original scale there is evidence to suggest patterns across each data set for Organisation Size. The central values would suggest that the organisation sizes increase with high-income developers rather noticeably. But this is not true for Education Level, where there is only a marginal increase in the central values. However, this could be due to a lack of sufficient samples to warrant greater diversity of Education Level across the clusters, and does not inherently mean that the category is an unsuitable characteristic.

Finally, the central Age values of each cluster can be observed. As should be expected, the patterns found in the Years Coding Professionally data is shared by the Age data. This is most evident in the overall increase in age for the Above-Median clusters, increasing from an average of 31.8 to an average of 38.2. And as with the Years Coding Professionally, the averages are being skewed by two particularly high clusters.

As an overview of this clustering, the process has clearly indicated unique characteristics and patterns that differ in the Above and Below Median income developers. These particular data columns should be continued in use for the next machine learning models as predictive values. The effectiveness of the clustering can be further evaluated by calculating the Silhouette Scores of each cluster. This score measures the density of the produced clusters, with scores closer to 1 indicating higher densities and better clustering. The score of the Below-Median data was 0.79, and the score of the Above-Median data was 0.77. These values are high enough to indicate that the clusters were dense and suggest that there are groupable data characteristics contained within the data set. This provides sufficient evidence to move forwards into attempting to produce machine learning models to predict high-income developers.

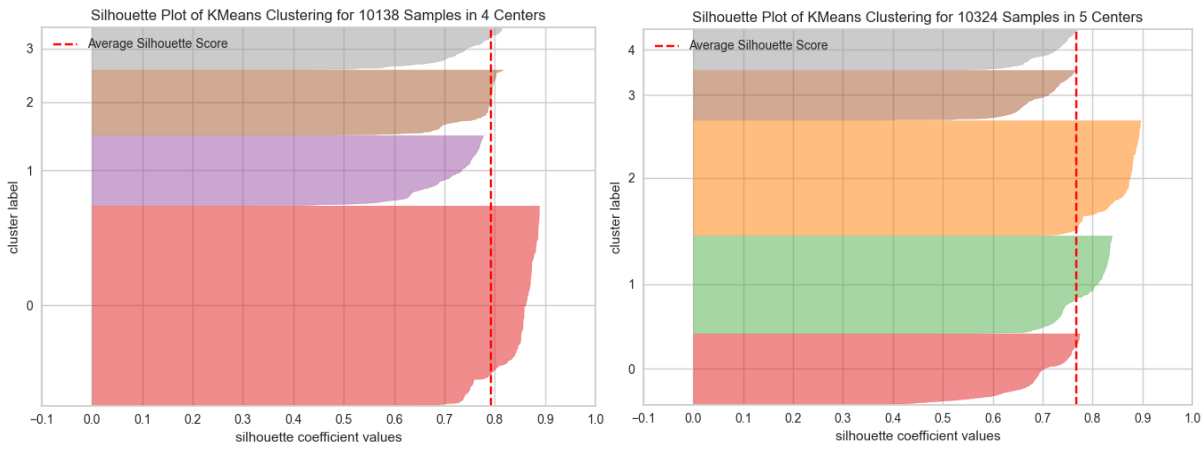

Figure 18: Silhouette Scores for the cluster groups of (Left) Below-Median Data and (Right) Above-Median Data

Machine Learning Methods and their Implementation

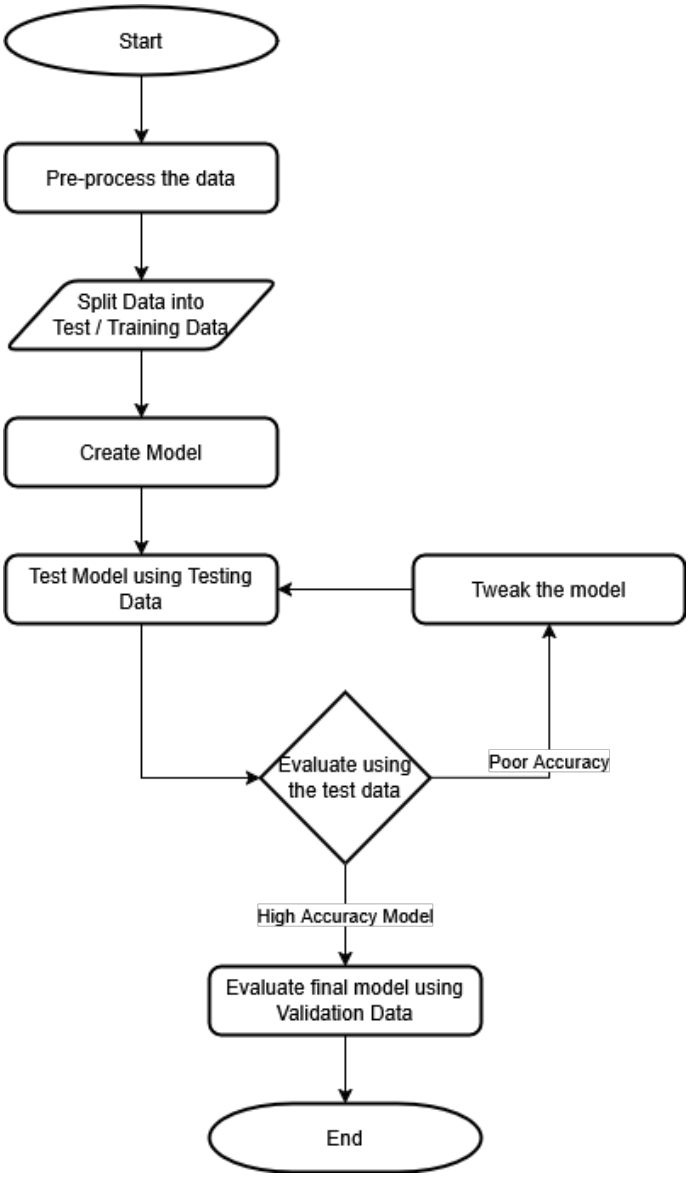

Figure 19: A flow diagram of the process of producing a Machine Learning model

Shown in Figure 18 is a diagram of the process flow of producing a Machine Learning model. First data is pre-processed, through cleaning and scaling of the values to ensure it will be suitable for a machine learning model. This transformation of the data was covered primarily by the previous section on Data Clustering.

The data can then be split into Training and Test data. The training data will be used to actively train the model, and the test data to validate the model's results. The test data can also be further split into validation data for final model testing and tuning. The separation of the data is necessary to ensure a good testing quality. Furthermore, it will also be used to prevent overfitting the model – making it too specific to the training data.

Then you will enter a process of continuously testing and tweaking the model until it meets some criteria set out – such as a particular accuracy/fit threshold. Or alternatively, multiple models may be produced and compared, with the most effective model selected and proceeded with. During this model development process, the model's hyperparameters will be the primary focus for refinement. Hyperparameters differ from standard model parameters as they are deliberately influenced by model trainers.

A final evaluation of the model could then also be performed using some additional Validation Data.

K-Nearest Neighbour

The K-Nearest Neighbour algorithm bases its output upon the values of the 'k' closest neighbouring values. A "voting" system can then be used to produce a singular output from the values of multiple different neighbouring values. However, for some classification situations, voting may end up in a tie. As a result, weighted voting may be an effective alternative – for example, the closer a neighbouring value is, the more its vote is worth. Fortunately, the goal of this project is the binary classification of developers into high or low income. As a result, any voting ties can be avoided by using an odd 'k' value.

The primary focus when tweaking the K-Nearest Neighbour algorithm will be the 'k' value, as there is no immediately obvious optimal value. Too small values would be faster but can lead to the model overfitting the training data. Alternatively, too large of a value may result in difficulties distinguishing data values as it will cause data values to become evened out.

The other hyperparameter to consider is the distance function for evaluating the similarity of two data points. Euclidian distance from $d(x, y) = p(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2$, will not necessarily be the same as the Manhattan Distance $d(x, y) = |x_1 - y_1| + \cdots + |x_n - y_n|$.

K-Nearest Neighbour also requires scaled numerical data in order to provide an accurate model. Hence, the transformations from the previous section of Cluster Analysis will be retained. Some further encoding of values for data columns will also be necessary, such as with the respondent's country. For this, frequency encoding has been used rather than individual columns. This may encourage some relationships between countries where less respondents have answered, but performance impact is expected to be marginal. The cause for not using Column-based encoding, is to prevent an excessive bloat in columns to analyse, and also due to KNN having difficulty in evaluating the distance between primarily binary 0 or 1 values – with comparison to a larger range of values. The respondent's Employment Status however – as it contains a list of categories - has been split into individual columns. This category would be difficult to only encode with frequency, as each respondent could select multiple options. Once this encoding is completed, the data subset can then be scaled to ensure an equal weighting in the model. Once the data is transformed, 20% of the dataset will be set aside as test data, with the remaining 80% used to train the model.

The first step that can be taken is attempting to determine the best parameters by using a randomized search from a set of potential parameters. Many randomized combinations of hyperparameters Is tested and their accuracy compared, with the most accurate model being used to determine the "best" parameters. However, as this is a randomized check, the results may not necessarily be the perfect values. As a result, we can do some additional hyperparameter testing, such as with the K-values. These can be increased or decreased, and the resultant accuracy graphed, ensuring that the best parameters have been chosen. The initial random search provided the hyperparameters of 91 for k neighbours, using the Manhattan distance approach, distance weighted values and the "kd-tree" algorithm.

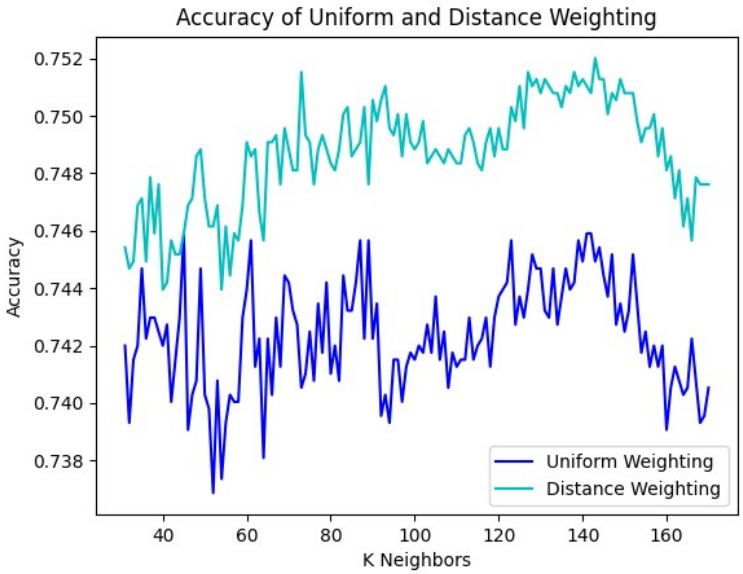

Figure 20: Accuracy of varying k values for Uniform and Distance weighted K-Nearest Neighbours Models

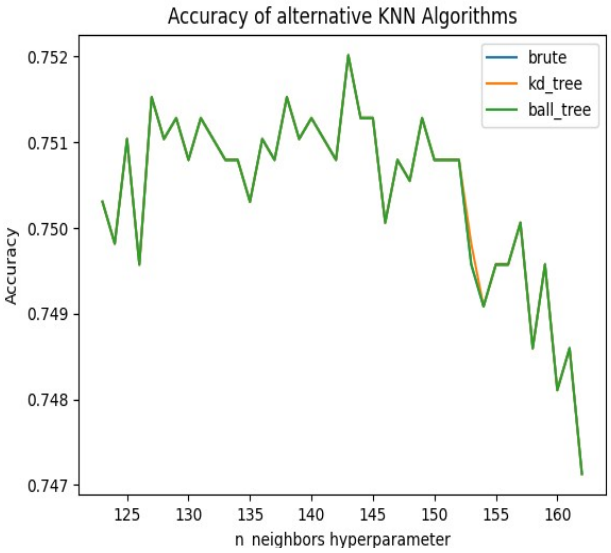

Figure 21: Accuracy of each Algorithm over varying k-values

To ensure that the optimal values have been chosen, additional comparative testing will also be done for the K Neighbours and alternative models. During this test in Figure 20, it was determined that the initial k value of 91 was not the most accurate, and instead 143, closely followed by a value in the 70s were more accurate. The weighting method does not change however, with Distance weighting consistently outperforming Uniform weighting.

Further testing can then be done with alternative algorithms to “kd-tree”. This will be centered around the updated k value of 143, due to it providing not only the highest accuracy, but also having a consistently higher accuracy within that region of +-10 k. The results however indicate that the algorithm has no effect upon the accuracy, with all reaching the same maximum accuracy of 0.752015636452480 to 15sf. As a result, the algorithm of “kd-tree” will be retained in the final model for K-Nearest Neighbours.

Decision Tree

The Decision Tree model is a collection of decision nodes. The initial and subsequent nodes of the tree will branch via a binary split into True and False paths, which eventually terminate in a leaf node. These leaf nodes are what determines the final classification of the input data. Decision trees are benefitted when the training data has a varied set of values, however, a lack of data for a particular category or subset will result in the tree having difficulties in accurately determining an output. In comparison to the prior K-Nearest Neighbour classifier, Decision Trees also tend to do better when there is a clear binary classification. They also tend to be quicker to train to fit data (de Ville 2013).

Although numerical data is not an outright requirement for decision trees as it is with K-Nearest Neighbour, the implemented models from Sklearn do require numerical data. As a result, all data will require transformation beforehand into numerical values. But in a difference to K-Nearest Neighbours, numerically labelled data can be used without issue of suggesting relational patterns – allowing Countries, for example, to be encoded as ordinal values 1, 2… rather than their frequency. Due to the nature of how the decision tree will use these values, the data set does not need to be scaled. Scaled or unscaled data will ultimately be treated the same as individual data values do not influence others via any weighting and are treated individually. Once the data is transformed, the same test/train split will be used. That is, 20% of the dataset will be set aside as test data, with the remaining 80% used to train the model.

The primary hyperparameters that will be considered for the decision tree will be the splitting criterion, and the maximum depth. For the split criterion, there are 3 methods that will be considered; ‘Gini’, ‘Entropy’ and ‘Log-loss’. The Gini Impurity method will calculate the probability that a feature is classified incorrectly when chosen at random. Features with the lowest Gini score are then used to build the final decision tree. The Entropy method attempts to measure the randomness contained within the data values. A lower entropy value indicates a homogenous set or subset of data, whilst higher values indicate a more scattered or random set of data. The entropy value for a feature can then be used to determine which features are used to build the decision tree. The Log-loss score is the final metric, which is used to build the decision tree that focuses upon the separation of groups or classes of data to improve its accuracy. As this is a binary classification issue, the classes involved are just above and below median income developers.

Initial hyperparameters can be determined with the use of a random search. This provides ‘Gini’ as the optimal split Criterion, with a max depth of 26 at an accuracy of 0.8041. From Figure 23, varying maximum depths and split criteria would be tested, leading to interesting observations. Firstly, the ‘Log-Loss’ and ‘Entropy’ criteria are completely identical in their accuracies, with ‘Gini’ being the overall most successful. Secondly, at a max depth of 19, the accuracy completely plateaus for the ‘Gini’ criterion, and plateaus at 22 for the other criteria. This plateau is likely caused by the decision tree not requiring any further depth or complexity, with the tree fully produced when reaching a depth of 19 or 22. Due to this reason, whilst the peak accuracy lies with the ‘Gini’ criterion at a depth of 13 (0.8031), it may be more suitable to use the depth of 19 as the model is fully produced with only a marginal affect upon the accuracy (0.8010) and may handle future data more effectively due to a larger tree.

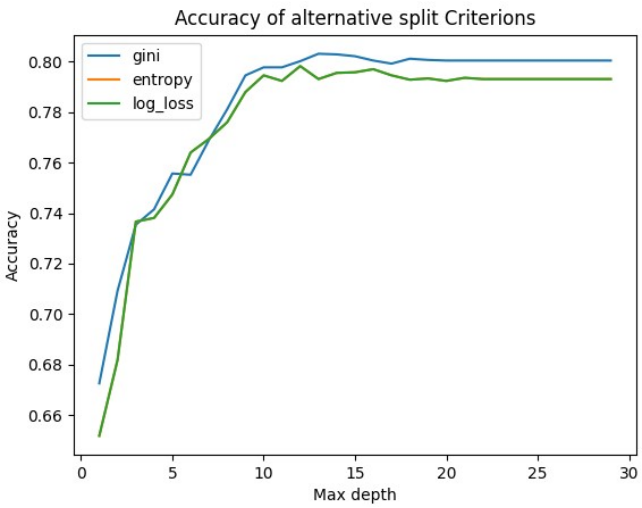

Figure 22: Accuracy of Alternative split criteria with varying max depth values

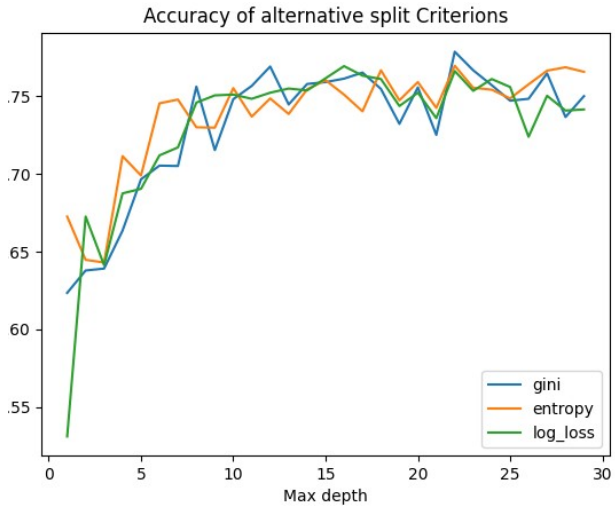

Figure 23: Accuracy of alternative split criteria as depth changes when using random splitting

The splitter is also used to determine which feature is used for building the decision tree. Using 'best' splitting, the feature with the highest importance calculated by one of the criteria methods will always be used. However, with random splitting, the feature is selected at random. The importance is used to determine how often the feature is selected instead.

The previous models in Figure 22 used the 'best' split, but when testing the 'random' split the model accuracy becomes significantly lower. It also becomes much more difficult to predict the accuracy of the various criterion models or make confident assessments of their accuracy patterns as depth increases. In figure 23, we can see that there is no plateau at the max depths around 20, and whilst the models may be increasing in accuracy marginally, it is difficult to predict if this trend would continue due to the randomness. Similarly, whilst the split criterions' influence is more evident within these models, it is difficult to determine any real superior criterion due to the randomness. Whilst the potential that a random model may produce a higher accuracy under the right random settings, these settings are particularly difficult to test. From the shown models, a peak accuracy was achieved by the 'Gini' criterion at 0.7786. As a result, the 'best' split method should be retained, due to the significantly higher predictability and accuracy metrics.

Random Forest

A random forest is an ensemble model of multiple Decision Tree algorithms. The decision trees are produced to fit various subsets of the provided data, allowing for better classification accuracy and more control over overfitting. A final classification is produced by using a voting method, whereby each tree effectively votes with their outputs for the final label (Breiman 2001). This is expected to show at least some increase in accuracy over the standalone decision tree model used previously.

As the model is an ensemble of decision trees, the requirements for data transformation and normalisation are the same. This will involve ensuring that numerically transformed data is used, without a requirement for scaling the data. Additionally, the label encoding will be sufficient

The key hyperparameters to tweak will involve the number of decision trees being used in the random forest, alongside the maximum features used when splitting a tree node. Generally, it can be expected that an increase in the number of trees will result in an overall increase in accuracy of the model. However, more trees result in a slower model, and increasingly reduced improvements. As a result, it will be important to determine a suitable number of trees for the forest, with testing ensuring that no further improvements can be made by increasing/decreasing the tree count. With regards to tree depth, this is shared by all trees within the forest.

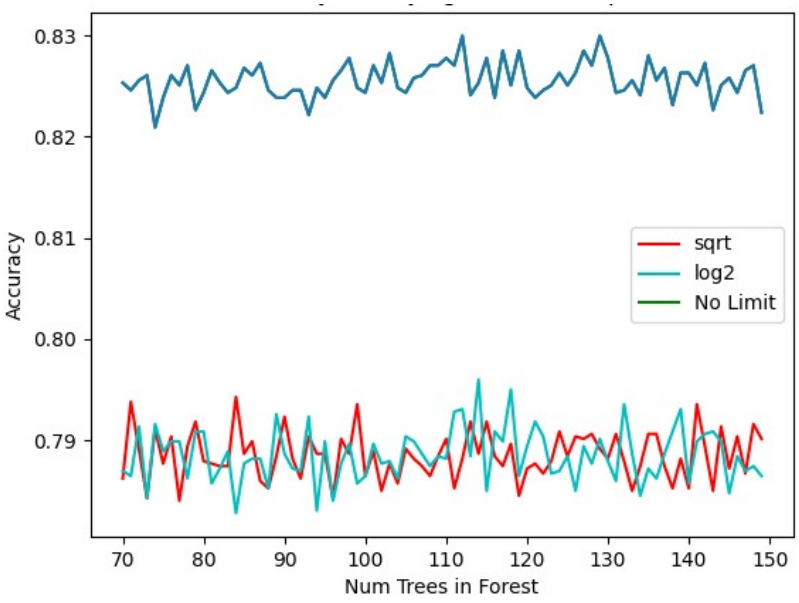

Figure 24: Accuracy of various tree counts, using alternative maximum feature counts (Despite the label, 'No Limit' refers to the blue line)

An initial random search produced 110 as the number of trees, whilst setting maximum features to be the square root of the total features involved. However this initial search only involved the 'sqrt' and 'log2' options, which resulted in a low predicted accuracy of only 0.7864 – which would be less than the singular Decision Tree. As this did not meet the initial expectations of a higher accuracy model, more testing was required. An uncapped number of maximum specifiers (labelled 'No Limit' in Figure 24) was revealed to be far more suitable than any restrictive approach, producing accuracies around 0.825, reaching a maximum accuracy of 0.8300 at 112 and 129 trees. Whilst there is the possibility that a higher accuracy model may exist with more trees, the models become too slow to produce and test beyond the 200 tree count. This slowness is also why the lower tree count of 112 will be proceeded with. With regards to the feature count, as no limitations proved itself vastly more accurate, the other two will be discarded.

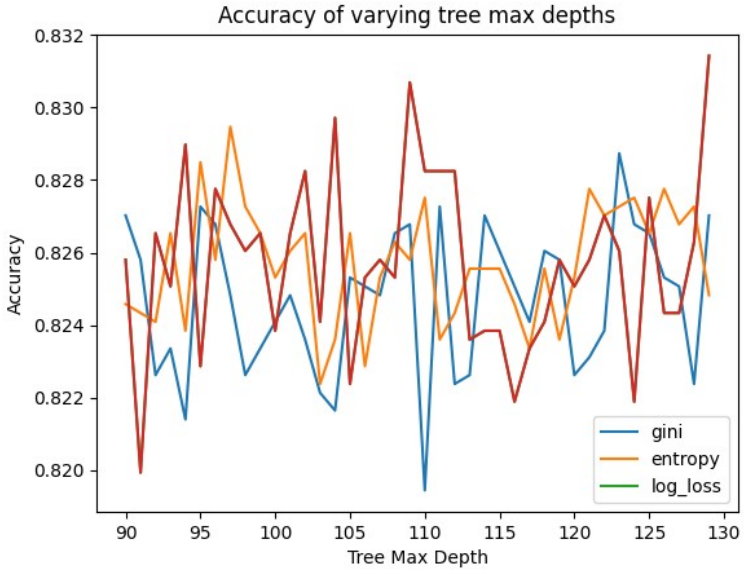

Figure 25: Tests of varying tree depths on each split criterion (Despite the label, 'log-loss' refers to the red line)

Another parameter which could be considered is the maximum depth and split criterions used in the decision trees, as was tested in the previous section with the singular tree. With regards to the split criteria, previous tests for the Decision Tree suggested the 'Gini' model was overall preferable. Yet with the Forest, 'Gini' became the lowest performing, with 'log-loss' achieving the highest accuracy peaks. Additionally, there is no noticeable plateau when increasing the maximum tree depth. Instead, the trade off to consider will be the increased time taken to produce and test models. The results of this testing indicates that the hyperparameters to proceed with are the 'log-loss' splitting criterion and a max tree depth of 129.

```
max accuracy:  0.8287319814317127 with depth:  123 with criterion:  gini
max accuracy:  0.8294649401417054 with depth:  97 with criterion:  entropy
max accuracy:  0.8314194967016858 with depth:  129 with criterion:  log_loss
```

Figure 26: The highest accuracy achieved by each criterion, and the max depth

Ensemble Model

Previously, each individual model used hyperparameter tweaking to find greater accuracies. Now that all 3 final models have been produced, an ensemble model can be made to potentially increase the prediction accuracy. Some of the methods for producing an ensemble model – such as Bagging – have already been used. The Random Forest model uses multiple decision trees by splitting the data into subsets and producing decision tree models for each subset. This model would also be the most successful, giving cause to using an ensemble approach. However, as the Random Forest model has been used, a slightly different ensemble method will be used.

One such ensemble approach that can be used is a Voting classifier. Rather than having multiple of a singular classifier upon subsets of the data, instead multiple different classifiers can be applied upon the whole data set. Each classifier will then produce an output label, and effectively votes for its output. Depending on the voting method, the results will then be determined from each vote by the models. In the current situation, only 3 models exist. As a result, no voting would be required, as a simple 2/3 majority would be all that is necessary. When using this approach, the aim is that uncertainty is reduced, by having multiple models predicting labels. However, if all models share the same flaws there is a concern that the ensemble model may worsen or at least retain this flaw. Previous observations noted that all models had difficulties in their accuracy in predicting High-Income developers specifically. Another concern arises from if K-Nearest Neighbours should be retained for an ensemble model. Having the weakest performance, if included it may detract from the overall quality.

Histogram Gradient Boosted Decision Trees (HGBT) is another ensemble model like Random Forest which employs Decision Trees. Due to this, it is expected to perform similarly well. HGBT employs a histogram-based algorithm to efficiently handle datasets of more than 10,000 samples – its speed making it preferential to a standard Gradient Boosting approach. Increasing amounts of trees can be added into the HGBT model to improve it, until further tree counts are negligible in effect. This model is also a self-improving model, using an iterative approach to make better models. This largely means that hyperparameter tweaking is done internally with each iteration, but there are still some observations that can be made. With regards to how the model develops its trees, it aims to reduce error using the Root Mean Square Error values calculated using $RMSE = \sqrt{\frac{\sum_{i=1}^{N} ||y(i) - \hat{y}(i)||^2}{N}}$.

Here, N is the total number of data points, with y(i) being the predictor values, and $\hat{y}$(i) being the corresponding prediction. The maximum number of iterations can then be tuned as a hyperparameter, as shown in Figure 27. This resulted in 115 as the value at which further iterations produced negligible change. With no further changes, the model would produce an accuracy of 0.8392, a new highest value.

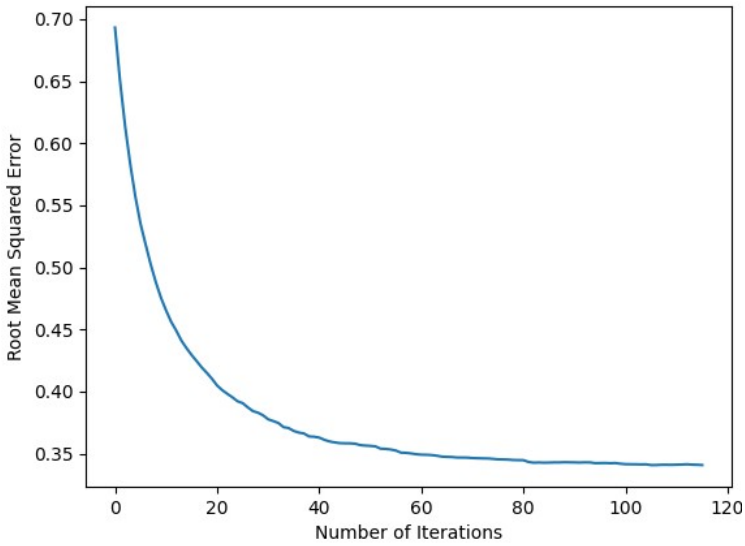

Figure 27: How increasing the number of iterations affects the RMSE score

Evaluation of the Machine Learning Models

For the evaluation of the finalised Machine Learning Models, first each model will be evaluated within its own merit. From there, it can be compared to the performance of the other models. A final evaluation can then be used to determine the best model. This final evaluation will use the same transformed data subset for each model, to avoid making changes that may explicitly favour or detriment particular models. This will be the frequency encoded data used initially with K-Nearest Neighbours.

Evaluating K-Nearest Neighbours

The primary hyperparameter which was tweaked was the K-value, determining the number of neighbours which would be checked for each data point. As previously covered, increasing the K-values typically saw correlation to higher accuracies. However, the models would become more computationally taxing as they checked more values, taking longer to compare. As a result, the value of 91 would be chosen for the number of neighbours – whilst not within the consistently higher region, it was faster and still reached almost the highest accuracy.

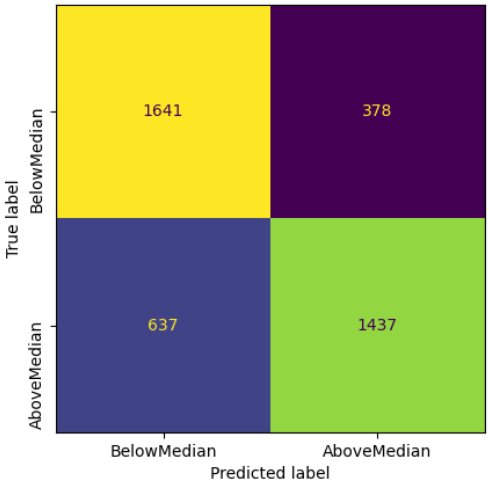

Figure 28: Confusion Matrix for the K-Nearest Neighbours Classifier. Numbers indicate amount of values for each category

Overall, the tested models would proceed to produce accuracies peaking at around 75% accuracy. The model was most effective at determining Below-Median income developers, having a specificity accuracy of 81.3%. But the model would largely be held back by its ability to predict Above-Median income developers correctly, reaching only 69.3% accuracy.

A further check on the accuracy score can also be done by varying the size of the training and test data. The results indicate that there have been no issues with regards to potential overfitting of the data as the training set size increases in proportion. However, at around 60% of the sample data size, the improvements become marginal. Other factors which may have affected the model's performance would likely have involved the data itself. The data was largely categorical, with a significant subset of it being binary 0 or 1 values. This is difficult to properly calculate distances between, and may have resulted in the lowered accuracy when compared to the other models.

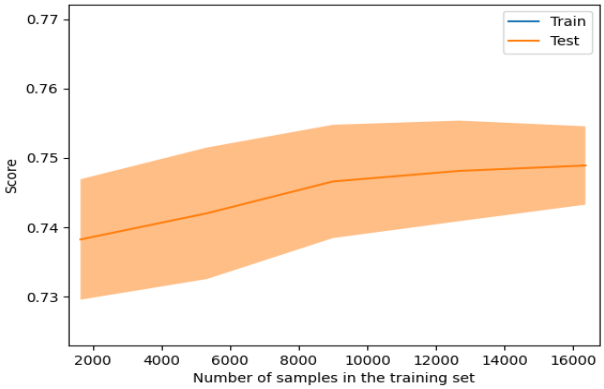

Figure 29: Accuracy Scores as the training data subset increases in size

Evaluating Decision Tree

The final decision tree produced used the 'Gini' Criterion whilst using the 'best' splitting option, and a max depth of 19. Some further parameters involved would also affect the minimum number of samples required to reach each leaf (classification) node, reducing the maximum complexity. These parameters would culminate in an accuracy of 80.1%.

Using the confusion matrix from the model, the predicted labels can be mapped against the true labels for each developer. This allows for a check on the accuracy metrics of true/false above and below median rates

to be calculated. Using the matrix, the overall accuracy is 80.2%, with 78.4% recall accuracy for high income developers and 82.2% specificity accuracy for Below-Median income developers. These accuracy results indicate that the decision tree is more accurate for predicting Below-Median income developers. It also predicted that developers would be below the median income 6.2% more often than above. This differs greatly to the actual data contents, as the actual proportion has High-Income developers being the largest category. Expectations would be that it over-predicted above-median income rather than below.

Despite this strange behaviour, it outperformed the K-Nearest Neighbours classifier by 5%. This is likely due to it handling the categorical data better, and not being affected by data scaling or outliers nearly as much.

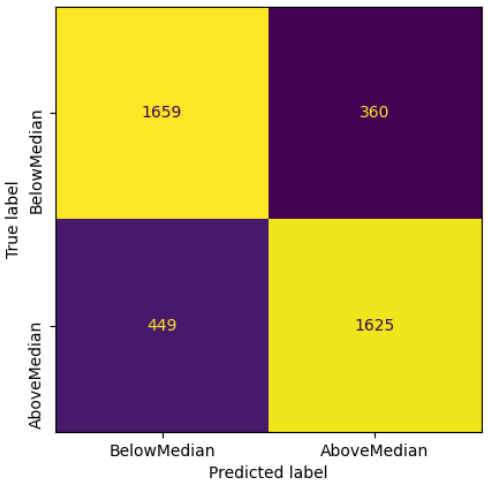

Figure 30: Confusion Matrix for the Decision Tree. Numbers indicate amount of values for each category

Evaluating Random Forest

Random Forest was expected to be the best of the initial 3 models, due to its implementation of multiple Decision Trees - which already had proven itself effective with the data set. And this would be the found results, as it attained the highest accuracy of the initial models at 83%. Since it was built using Decision Trees, it can also be expected that some patterns observed during the Decision Tree evaluation may still appear – such as over-predicting Below Median.

Using the confusion matrix, we find an overall accuracy of 82.9%, with a recall accuracy of 80.4% for above-median income developers, and a specificity accuracy of 85.4% for below-median income developers. And with this model, there is actually an 8% increase in the amount of Below-Median predictions compared to Above-Median. This is the same trend as with the Decision Tree, however it is not evidence that increasing the amount of predictions for the Below-Median category increases the overall success rate.

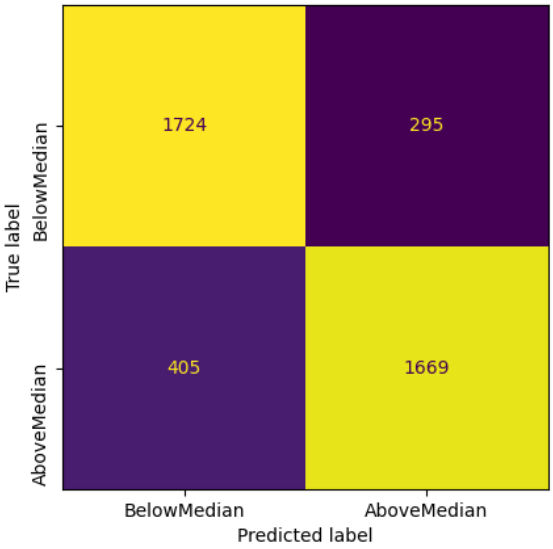

Figure 31: Confusion Matrix for the Random Forest. Numbers indicate amount of values for each category

Initial Model Comparison and Evaluation

Further evaluation and comparison of the models can be performed using a Receiver-Operating Characteristic curve. A perfect classifier would have a 1.0 True Positive rate, with a 0.0 False Positive Rate. As a result, the overall success of a model can be measured by how close it gets to this perfect classifier. The area under the curve (shown as 'auc' in the diagram legend) is calculated to give a more precise numerical value for determining the best accuracy. A purely random classifier is expected to achieve 50% accuracy and is plotted as the dashed line.

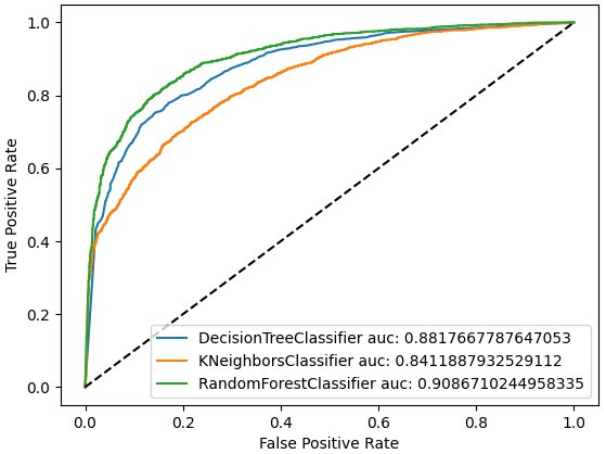

Figure 32:ROC Curve of the initial 3 models

In Figure 32, a comparison of the original 3 models' accuracies upholds the Random Forest as the most effective classification model for the provided data. However, when comparing the Confusion Matrices, there are important characteristics that should be addressed. First, the K-Nearest Neighbours Classifier was close in accuracy in predicting Below-Median developers to the Decision Tree. And so whilst the model's overall success was lower, it was not as poorly performing as initial accuracy insights may suggest. The other characteristic was that all models were overall less successful in predicting High-income developers when compared to predicting those earning Below the median. This does align with the findings during cluster analysis, where the Below-Median data produced a higher average silhouette score. These continued observations suggest that overall there are less common characteristics or data patterns contained within the high income developer subset.

Evaluating the Ensemble Models

The first ensemble model that will be produced is a Voting Classification model due to its simplicity to implement. Collating the models of K-Nearest Neighbours, Decision Tree and Random Forest results in a classifier model which reaches a consistent 82% accuracy. An initial concern was that the lower accuracy of the K-Nearest Neighbours classifier may result in an overall reduction in the capabilities of a voting model. As a result, the impact of removing a particular model has been measured against the initial all-model voting performance to determine each model's impact.

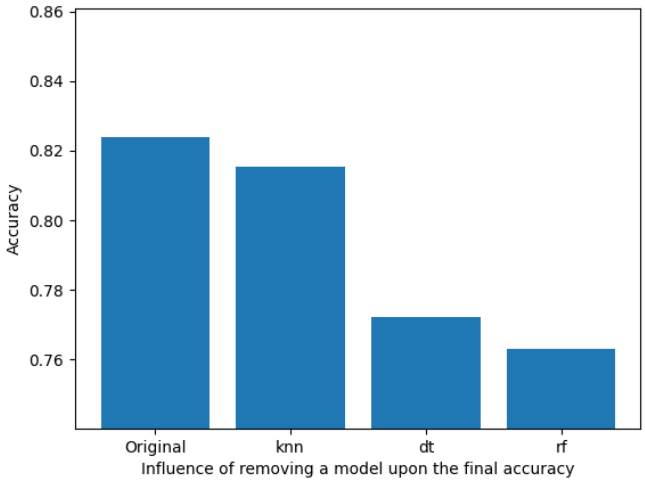

Figure 33: Changes to accuracy when the specified model is removed from the Voting Classifier

The K-Nearest Neighbours model is evidently the least impactful, making the smallest reduction when removed from the model. However, it does remove any doubts about it being detrimental to the overal model success, as the Original model is left with the higher accuracy. What does become evident is that the Decision Tree and Random Forest are by far the most impactful, as removing those estimators results in a reduction in accuracy by 5% and 6% respectively. And this is further apparent when observing the Confusion Matrix. It is nearly identical to that of the Random Forest, differing by only 9 more values in both the Above Median and Below Median categories being predicted incorrectly. The overall results from this model are not sufficient enough to suggest that it would perform consistently better than the Random Forest on its own.

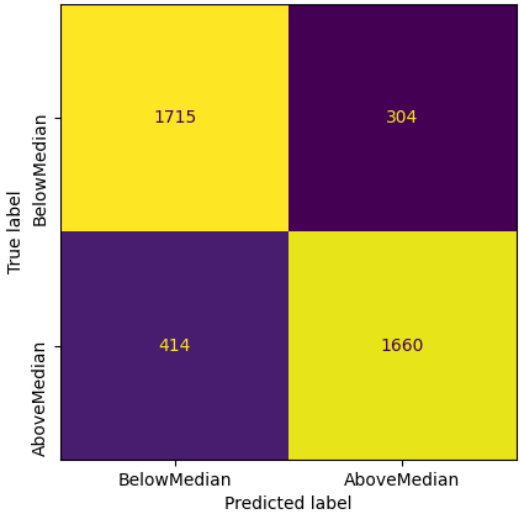

Figure 34: Confusion Matrix for the Voting Classifier

The other ensemble model to analyse is the Histogram Gradient Boosting Decision Tree classifier. As the model uses multiple decision trees like the Random Forest classifier, this also means that some hyper parameters will affect the trees themselves. During the tree-development process, each iteration largely will resolve issues with hyperparameters internally. However, we can still produce a graph to observe the influence of these hyperparameters upon the model's accuracy. The accuracy peaks at 84.1% when using a maximum depth of 8, and plateaus at a depth of 19 – the same as the original Decision Tree model.

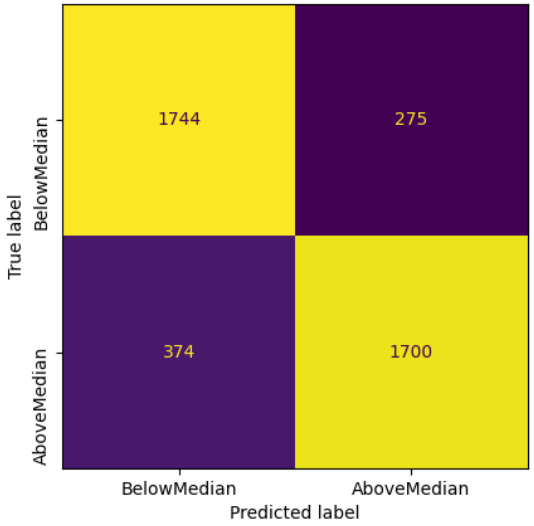

Figure 35: Confusion Matrix for the HGBD

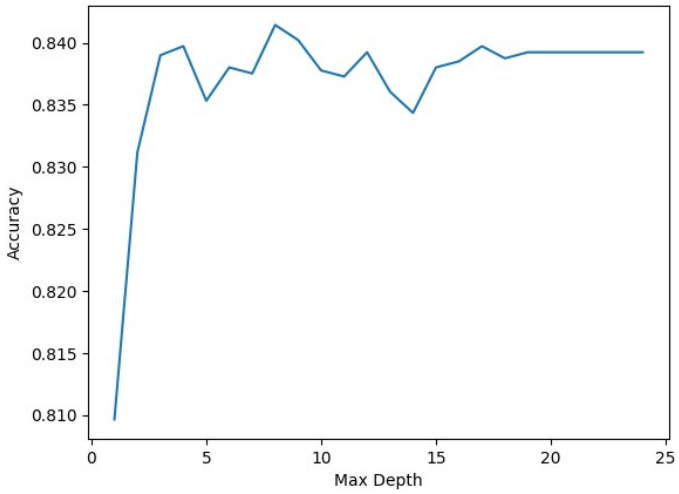

Figure 36: The influence of altering the max tree depth upon the HGBD, which initialises with an unlimited max depth

The confusion matrix for this classifier follows the same patterns as before, being weakeset in its classification of Above-Median income developers. However, it does also conclude that this ensemble model is the most accurate in all categories. Capable of 82.0% recall accuracy in predicting high income developers, and 86.4% specification accuracy when predicting Below-Median developers, it is evident that the ensemble approach has proven itself to be the most successful. In comparison to the Random Forest classifier, this model observed an increase in recall by 1.6% and an increase in specification by 1%. And in comparison to the Voting classifier, this model observes an overall increase in accuracy of 2%.

Final Model Evaluation and Comparison

When creating an updated ROC curve, the prior findings are once again reinforced. The ensemble models are the most successful, HGBD being the most successful, with Random Forest Classifier and the Voting classifier being very close in their accuracy. The results indicate that the data overall was most suited to models which did not employ distance-based metrics, and could deal with binary or categorical metrics effectively. However, despite being the weakest model, the K-Nearest Neighbours model did still retain an accuracy score of around 75% - or 0.84 if using the ROC metrics. And so it can be deemed as a successful model, but it was heavily outperformed.

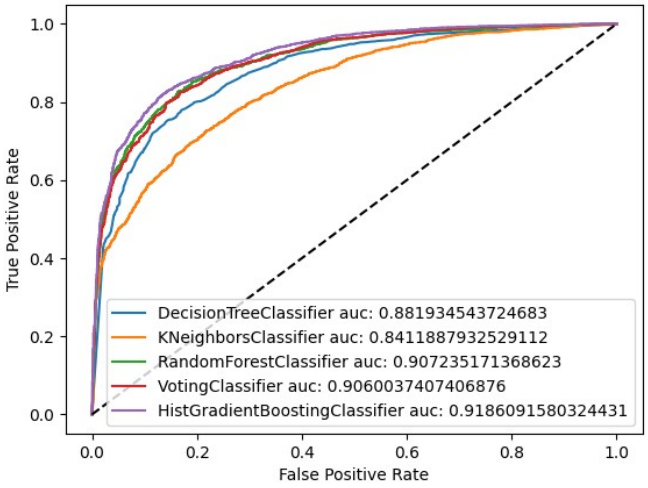

Figure 37:An Updated ROC curve to include the new Ensemble Classifiers

Final Accuracy Results for the produced Classification Models	
Histogram Gradient Boosted Decision Tree	84.1%
Random Forest	83.0%
Ensemble Voting	82.1%
Decision Tree	80.1%
K-Nearest Neighbours	75.0%

Conclusion

The initial objective of this report was to produce a classification model that could predict high-income developers based upon their characteristics. Through the Histogram Gradient Boosted Decision Tree model, this objective has been achieved, attaining an overall accuracy score of 84.1%. This is a significant improvement over randomly guessing by 34% and shows that high-income developers have characteristics and patterns that can enable them to be predicted. With regards to the methodology, CRISP-DM proved itself beneficial as it was applied throughout the proceeds of this report. The exploratory data analysis provided insight into some useful categories and trends that could be followed up on with the produced clusters and models – for example, how an increase in ages correlated to increased median wages. Using these established foundations and early data cleaning resulted in success when attempting to determine clusters within the data.

However, when reflecting, the processes do have some areas that could be improved. Firstly, the lower silhouette score for high-income developers during cluster analysis could have been taken as a sign that the selected Characteristics were sub-optimal. These characteristics would proceed to affect the produced models, as all models shared their issues in predicting high-income developers when compared to their accuracy in predicting below-median income developers. This also did not appear to be a failure with decision trees either, as the K-Nearest Neighbours also saw disproportionate accuracy. And so the first suggestion would be to review data clustering with new observational values – if necessary producing a model specialised to predicting high-income over below-median income developers. This could then be used in a voting classifier or similar ensemble approach. The second suggestion would be to attempt alternative models to Decision Trees. Decision Trees saw evident success with this data, which resulted in their continued use in ensemble models. However, it would be worth exploring other classification options.

My Development in Machine Learning

For my personal development, this report has provided me with multiple insights. Firstly, the benefits of extensive testing and tuning. Modifying the hyperparameters and evaluating the resultant accuracies of each model made it evident as to why the process is necessary, and gave further insight into how models adapt to their training data. My understanding has also been improved by learning how models have their own individual strengths and weaknesses, which extends into how models may need different approaches when encoding data. For example, the K-Nearest Neighbours model was poorly suited to a binary classification of data and could potentially have been more suited to predicting continuous values. Similarly, it was also subject to attempting to produce relations between label encoded data such as countries – which the later Decision Tree models did not suffer from. As a result of this learning, for future projects I will attempt to better choose models more suitable for the data set.

The overall process has also given insight into how machine learning approaches can be applied in not only a research setting, but also within a business context. Applying machine learning models to collected user data could be used to make predictions that may be beneficial to the business. For example, categorising users based upon shared characteristics would be useful for targeting things like advertising. As a result of this process, I now have far greater understanding of the importance that AI processes like machine learning hold for the field of data science.

Appendix

Selected Data Features

Name	Description	Data Type	Example Data	Encoding Method	Reason for Selection
Age	What age group the respondent belongs to.	Categorical (Ordinal)	'35-44 years old'	Label Encoding: Midpoint age of each group used as the value	Shown during Exploratory Data Analysis, as the age group increases, the proportion of repondents earning above the median income increases.
Organisation Size	Estimated count of people in respondent's organisation	Categorical (Ordinal)	'20 to 99 employees'	Label Encoding: Midpoint organisation size of the group used as the value	Whilst a large organisation size may not correlate to higher wage, it can be expected that the smaller organisation sizes will generally correlate to lower wage.
Education Level	Highest achieved level of education	Categorical (Ordinal)	'Master's degree (M.A., M.S., M.Eng., MBA, etc.)'	Ordinal Encoding: values 0-7 increasing as higher education achieved	It is expected that individuals with a higher education (eg: Masters/PHD) would have a higher income
WorkingRemotely	Regards if the respondent works in-office, hybrid or remote	Categorical (Ordinal)	Hybrid (some remote, some in-person)'	Ordinal Encoding: values 0-2 increasing as developer becomes more in-office	It is expected that there may be a slight decrease in pay when the respondent has a fully remote role, which may affect the proportion of remote workers in the high-income category
Purchase Influence	How much influence on company purchases the respondent has	Categorical (Ordinal)	I have some influence'	Ordinal Encoding: values 0-2 increasing as Influence increases	Respondents with higher purchase influence are likely in higher positions, resulting in increased wages. Hence it is expected that this leads to more being in the high-income category
Years Professional Coding	How many years the respondent has been coding professionally	Discrete Numerical	16	No encoding required, values are already numerical	As shown during EDA, Industry appeared to affect the years of code experience required to reach a high-income status, so this data will be used for the models
Country	The country the respondent is from/resides in primarily	Categorical (Nominal)	Pakistan'	Frequency Encoding: values assigned from frequency of Country in responses	It is expected that the respondent's country / location may influence how much they earn. For example, developers in the US could earn more than those in the UK, making them more likely to be high-income developers
Developer Type	The respondent's primary role	Categorical (Nominal)	Data engineer'	Frequency Encoding: values assigned from frequency of Country in responses	Senior developers can be expected to earn more than junior developers. As a result, it is expected that there will be differences and patterns in what roles have larger/smaller proportions of high-income developers
Industry	What industry the respondent works in	Categorical (Nominal)	Fintech'	Frequency Encoding: values assigned from frequency of Country in responses	As shown during EDA, Industry appeared to affect the years of code experience required to reach a high-income status, so this data will be used for the models
Employment Type	The current employment or student status of the respondent	Categorical (Nominal) (Listed)	Employed, part-time'	One-Hot Encoding: new columns produced for each listed employment type	Generally, part-time employees will be earning less than full-time employees. And so this data will also be included in the final selection

References

The full list of references used within this report is as follows:

Breiman, L., 2001. Random Forests. *Machine Learning,* 45, p. 5–32 [online]. Available at: https://doi.org/10.1023/A:1010933404324 [Accessed: 28 March 2025].

Chapman, P., Clinton, J., Kerber, R., et al., 1999. The CRISP-DM user guide. *4th CRISP-DM SIG Workshop in Brussels in March* [online]. Available at: https://s2.smu.edu/~mhd/8331f03/crisp.pdf [Accessed on: 13 March 2025].

de Ville, B., 2013. Decision trees. *WIREs Computational Statistics,* 5 (6), pp. 448–455 [online]. Available at: https://doi.org/10.1002/wics.1278 [Accessed: 28 March 2025].

Huber, S., Wiemer, H., Schneider, D., et al., 2019. DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model. *Procedia CIRP,* 79, pp. 403–408 [online]. Available at: https://www.sciencedirect.com/science/article/pii/S2212827119302239 [Accessed: 14 March 2025].

J. S. Saltz, 2021. CRISP-DM for Data Science: Strengths, Weaknesses and Potential Next Steps. In: *2021 IEEE International Conference on Big Data (Big Data),* pp. 2337–2344 [online]. Available at: https://ieeexplore.ieee.org/abstract/document/9671634 [Accessed on: 13 March 2025].

Saltz, J., Hotz, N., Wild, D., et al., 2018. Exploring project management methodologies used within data science teams [online]. Available at: https://aisel.aisnet.org/amcis2018/ITProjMgmt/Presentations/12 [Accessed: 10 March 2025].

Schröer, C., Kruse, F., and Gómez, J.M., 2021. A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science,* 181, pp. 526–534 [online]. Available at: https://www.sciencedirect.com/science/article/pii/S1877050921002416 [Accessed: 10 March 2025].

Stack Overflow, 2024. *Stack Overflow Developer Survey* [online]. Available at: https://survey.stackoverflow.co/ [Accessed 15 March].

Wirth, R. and Hipp, J., 2000. 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,* Manchester, pp. 29–39 [online]. Available at: https://www.cs.unibo.it/~danilo.montesi/CBD/Beatriz/10.1.1.198.5133.pdf [Accessed on: 13 March 2025].