MACHINE LEARNING AND AI FOR DATA ANALYSIS

Abstract

This report presents a comprehensive exploration into the use of machine learning algorithms for predicting high-income brackets within a developer community. Through meticulous analysis and application of various classification models, including k-Nearest Neighbors (k-NN), Random Forest Classifier (RFC), Gradient Boosting Machines (GBM), and Neural Networks, we aim to unveil the intricate relationship between a developer's background, coding experience, and their income level. Special attention is given to the process of hyperparameter tuning and ensemble methods to enhance model performance. Further, we delve into cluster analysis to categorize developers based on similar attributes, providing insights into the diverse profiles within the tech industry. The study concludes with a critical evaluation of each model's accuracy through confusion matrices, ROC curves, and the Area Under the Curve (AUC), offering a nuanced understanding of their predictive capabilities. Our findings not only contribute to the academic discourse on predictive modeling but also offer practical implications for targeted educational programs and hiring strategies in the tech sector.

Awn Albqowr 2022 (N1138321)

N1138321@my.ntu.ac.uk

Contents

Section 1- Introduction to Machine Learning Coursework

Artificial Intelligence (AI) and Machine Learning (ML) have become essential in the interpretation of complicated data sets to extract relevant insights in an era where technological prowess is paramount. This course, which is a part of the extensive module "Foundations of Artificial Intelligence and Machine Learning," is an exploration into the real-world applications of AI and ML ideas, not just a requirement for the classroom. The main objectives of this coursework are to map out the landscape of the 2021 freeCodeCamp New Coder Survey data set through a thorough exploratory data analysis (EDA), identify underlying coder archetypes through a careful cluster analysis, and carefully create classification models that predict income brackets based on a variety of attributes.

We hope to learn a great deal from these assignments. The goal is to learn more about the demographic, educational, and financial backgrounds of those who are pursuing careers in coding, rather than just finishing a required course. Essentially, the coursework acts as a prism through which we may see the goals, difficulties, and accomplishments of the younger generation of programmers, who will play a significant role in determining the direction of the computer sector. In addition to their academic value, we hope that these insights will serve as a basis for developing strategies that will help decision-makers in the education, government, and business sectors.

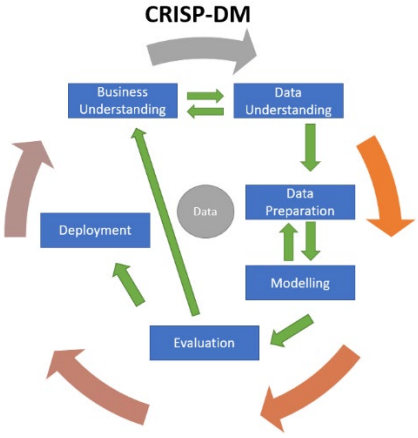


Figure 1 - CRISP-DM Cycle

At the heart of this empirical investigation is the Cross-Industry Standard Process for Data Mining (CRISP-DM), a well-established, robust methodology that provides a systematic framework for data analysis projects. CRISP-DM's iterative approach involves six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. This methodology is particularly crucial to the project as it instills a structured discipline that is vital for navigating the complex, often chaotic landscape of raw data.

Using CRISP-DM for this project, we start by developing a thorough **understanding of business and data**, which is the point at which our academic quest and practical inquiry converge. In this stage, we precisely outline our goals in order to disentangle the complex elements influencing the pay scales of novice programmers, with the ultimate goal of clarifying the underlying stories embedded in the data. We look beyond the obvious in search of trends and connections that highlight the complex interactions of educational attainment, technical proficiency, and financial results. We approach the data understanding process with great care and attention to detail. By using exploratory data analysis (EDA), we are able to map out the features in the dataset and their interactions with one another quite precisely. Layer by layer, it's a careful investigation to ensure that our subsequent analysis is built upon a solid, empirically-grounded foundation.

During the **data preparation** stage, we undertake the extensive process of carefully and methodically going through the dataset. In order to ensure that the data is in perfect condition for the upcoming analytical work, we roll up our sleeves and get down to business at this point. We separate the most important characteristics from the clutter of irrelevant information, concentrating our attention on factors that have a strong ability to predict income levels. This selection process is anything but random; rather, it is a deliberate decision-making exercise, emphasized by a careful examination of the details of the data and a methodical approach to determining what is most important. It is here that we apply critical thinking and domain expertise to discern the attributes that will serve as the keystones in our predictive models.

In the **modeling phase**, we test the theoretical knowledge acquired in the classroom against the real-world complexities of the freeCodeCamp New Coder Survey dataset. Our first foray is into cluster analysis, where we unveil the inherent groupings among new coders, each cluster potentially unraveling a different story within the data. Building on this, we develop classification models with the ambition to accurately predict developers' income levels. The selection of our models is judicious, drawing on a diverse set of algorithms that promise both robustness and insight. Their implementation is an intricate dance of precision, involving the careful adjustment of parameters to refine the models until they echo the truths hidden within the data. This phase is a true testament to our analytical rigor and our commitment to extracting meaningful predictions from complex data.

The **evaluation phase** stands as a critical checkpoint in the data mining process, a juncture where the robustness of our machine learning models are rigorously scrutinized. Through the application of various performance metrics such as accuracy, precision, recall, and the nuanced interpretations of the ROC curve, we assess the predictive power of our models. This comprehensive evaluation goes beyond mere numbers; it involves a thoughtful analysis of the models' ability to generalize to unseen data, thus ensuring their practical applicability.

In this crucial phase, we not only affirm the statistical significance of our results but also consider their relevance in the context of the real-world scenarios they aim to represent. This is not simply a

procedural step to be checked off; it is a deliberate and reflective process that confirms the reliability and validity of our analytical outcomes. By carefully examining the strengths and potential weaknesses of our models, we calibrate and refine our algorithms, optimizing them to better capture the complexity of the data. This iterative process embodies the scientific rigor essential to the field of machine learning, anchoring our conclusions in a foundation of empirical evidence and methodological soundness.

In the **deployment phase**, we hypothetically explore the practical application of our coursework findings. Though direct deployment is not an immediate step within an academic setting, we consider the far-reaching implications of our analytical models. How might these models serve educational institutions, guiding curriculum adjustments to better align with the evolving demands of the technology job market? Or in the hands of recruiters, these insights could streamline the identification of promising candidates, optimizing the match between a coder's skill set and job requirements. Envisioning such applications, we bridge the gap between theoretical analysis and potential real-world impact, acknowledging the influence that thoughtful deployment of these models could have on shaping the pathways to success for new coders. [Nick Hotz, 2018]

By embarking on this journey through the CRISP-DM framework, we not only enhance our understanding of machine learning but also embark on a path of discovery that extends well beyond the confines of the classroom. This coursework represents a synthesis of academic learning and practical application, resulting in a comprehensive analysis that is as enlightening as it is educational.

Section 2 - Data Understanding, Data Pre-Processing, and EDA

Data Collection Method

The survey was conducted online, leveraging freeCodeCamp's extensive reach within the coding and technology learning community. Participants likely accessed the survey through direct links shared on freeCodeCamp's website, social media platforms, email newsletters, and possibly through other coding-related forums and platforms. The broad range of questions, designed to be inclusive of diverse experiences and backgrounds, indicates a comprehensive approach to gathering data from individuals at various stages of their coding journey, from beginners to those already employed in tech roles.

Purpose of the Data

The data collected through this survey serves several key purposes:

1. **Understanding Learning Pathways**: Questions about reasons for learning to code, methods used for learning, and resources found helpful aim to uncover how individuals are

navigating their coding education. This can inform freeCodeCamp and similar platforms on how to better support learners through content, tools, and community engagement strategies.

2. **Career Aspirations and Employment**: By asking about employment status, interest in software development careers, and expectations for future jobs, the survey seeks to understand the career aspirations of new coders. This information can help tailor educational offerings to align with the skills and knowledge most relevant to the tech industry's needs.
3. **Demographic Insights**: Questions on age, gender identity, ethnicity, geographic location, and educational background help paint a comprehensive picture of the coding community's diversity. This demographic information is crucial for identifying gaps in access to coding education and opportunities, aiming to make learning more inclusive and accessible.
4. **Economic Impact**: Inquiring about expenses related to learning to code, expected earnings, and current financial situations allows for an assessment of the economic impact of coding as a career path. It also sheds light on the financial barriers that learners might face.
5. **Lifestyle and Preferences**: Questions about preferences for remote work, willingness to relocate for jobs, and satisfaction with various aspects of current employment offer insights into the changing landscape of work, especially in the context of the tech industry. This can guide companies and educators in adapting to the preferences and expectations of the new workforce.

Important Data attributes

No	Attribute	Description of Attribute	Related Questions	Numeric/Descriptive	Data Type
1	WeeklyHoursSpent	The number of hours the respondent spends learning to code each week	7) About how many hours do you spend learning each week?	Numeric	Float
2	MonthsProgExp	The total number of months the respondent has been programming	8) About how many months have you been programming?	Numeric	Float
3	EduLevel	The highest level of formal education that the respondent has achieved	32) What is the highest degree o level of school you have completed?	Descriptive	Object
4	EmploymentStatus	Whether the respondent is currently employed in a software development job	10) Are you already employed in a software development job?	Descriptive	Object
5	AnnualIncomeUSD	The total amount of money earned by the respondent in the last year from any job or employment, expressed in US Dollars.	22) About how much money did you earn last year from any job or employment (in US Dollars)?	Descriptive	Object
6	Age	How old the respondent is.	23) How old are you?	Numeric	Float
7	InterestInDevCareer	Signifies whether the respondent, who is not currently a developer, is interested in pursuing a career in software development.	12) If you are NOT already a developer, are you interested in a software development career?	Descriptive	Object
8	Region	Tells us which region the respondent is from.	26) Which part of the world do you live in?	Descriptive	Object

Justification for the chosen attributes

1. **WeeklyHoursSpent (Float)**

    - **Justification**: The commitment to learning, as measured by hours per week, is a strong predictor of skill acquisition speed and depth, directly influencing employability and income potential. This feature is essential for exploratory data analysis to identify trends and for classification models to predict income levels,

aligning with the coursework's objectives of understanding the impact of learning behaviors on career outcomes.

2. **MonthsProgExp (Float)**

    - **Justification**: The duration of programming experience is crucial for measuring a learner's proficiency and market readiness. It's vital for both exploratory analysis, to explore the relationship between experience and income, and in classification tasks, serving as a predictor of income bracket. This aligns with the data understanding and modeling phases of CRISP-DM, emphasizing the role of experience in career development.

3. **EduLevel (Categorical)**

    - **Justification**: The educational background can influence access to certain job markets and salary levels. For cluster analysis, it helps in identifying segments within new coders based on their educational attainment and potential income disparities. It's a critical factor in the coursework's focus on analyzing educational impact on coding careers, fitting into the data preparation and exploratory analysis stages.

4. **EmploymentStatus (Categorical)**

    - **Justification**: Being employed in software development is a direct indicator of a learner's transition into the tech industry. This feature is used for segmentation in cluster analysis and as a categorical predictor in income classification models, aligning with the project's aim to assess employment outcomes of coding learners.

5. **AnnualIncomeUSD (Categorical)**

    - **Justification**: Incorporating "AnnualIncomeUSD" as a predictor offers significant insights into the financial benefits of coding proficiency, directly linking to the coursework's focus on the economic impact of coding skills. Its inclusion provides a valuable perspective on the earning potential of new coders, essential for income estimation. The rationale behind selecting this attribute aligns seamlessly with the coursework's objectives, which aim to unravel the dynamics between demographic factors and income. Utilizing this attribute complements the CRISP-DM methodology by elucidating the economic ramifications of coding education, underscoring the importance of practical skills in the modern workforce and their correlation with financial success.

6. **Age (Float)**

    - **Justification**: Age is a crucial predictor that may reflect career stage and experience, relevant for income estimation. Its inclusion aligns with the coursework's objective to explore demographic factors influencing income. In modeling, age can help distinguish between entry-level and seasoned professionals, aiding in the prediction of income levels. This aligns with the CRISP-DM methodology by offering insights into the age distribution of new coders and its

correlation with their earning potential, highlighting the coursework's focus on the economic aspects of coding education.

7. **InterestInDevCareer (Categorical)**
    - **Justification**: This attribute could be particularly useful for understanding the motivations of those learning to code and could provide valuable insights when cross-referenced with other data, such as the amount of time they spend learning each week.

8. **Region (Categorical)**
    - **Justification**: One important aspect to understand the complex influences on coding education and employment is the 'Region' attribute. It clarifies regional differences by showing how a person's location affects their ability to access resources for education, the state of the economy at large, the demands of the local market, and cultural perceptions about employment in technology. This characteristic helps map the global distribution of developers and tech sector trends, offering insights into local skill needs, the dynamics of remote work, the influence of regulations, and the possibility of focused training initiatives. 'Region's' inclusion is in line with the objective of investigating how socioeconomic factors influence a developer's career path and how these elements link to educational attainment and earning potential.

Data Cleaning

Data cleansing is a crucial phase in the data analysis pipeline, integral to enhancing the quality of the data and, consequently, the insights derived from it. The process involves detecting and amending inaccuracies, gaps, and inconsistencies in the data set to ensure that subsequent data analysis stands on a foundation of accuracy and reliability.

In the context of our current dataset, a preliminary examination reveals a substantial amount of missing data across various attributes (as shown in Figure 2). Missing values can arise due to a multitude of factors, such as errors during data collection, non-responses on surveys, or even during data entry and transmission. It's imperative to handle these missing entries with meticulous care since they can potentially bias the analysis, leading to skewed results or erroneous interpretations.

[4]:

	WeeklyHoursSpent	MonthsProgExp	EduLevel	EmploymentStatus	AnnualIncomeUSD	Age	InterestInDevCareer	Region
0	4.0	120	Professional degree (MBA, MD, JD, etc.)	Yes	NaN	33.0	NaN	Europe and Central Asia
1	10.0	6	Bachelor's degree	No	Under $1,000	38.0	I am already a developer	Latin America and Caribbean
2	30.0	48	Master's degree (non-professional)	No	NaN	NaN	Yes	North America
3	NaN	36	High school diploma or equivalent (GED)	No	NaN	19.0	Yes	Sub-Saharan Africa
4	2.0	24	Bachelor's degree	Yes	40,000to49,999	35.0	I am already a developer	East Asia and Pacific
...	...	...	...	...	...	...	...	...
18121	40.0	12	Master's degree (non-professional)	No	NaN	44.0	Yes	Middle East and North Africa
18122	NaN	NaN	Bachelor's degree	No	NaN	37.0	Yes	North America
18123	10.0	13	Associate's degree	No	20,000to24,999	23.0	Yes	North America
18124	6.0	36	Professional degree (MBA, MD, JD, etc.)	Yes	30,000to34,999	34.0	I am already a developer	Europe and Central Asia
18125	5.0	200	Master's degree (non-professional)	No	10,000to14,999	44.0	No	Europe and Central Asia

18126 rows × 8 columns

Figure 2 - Dataset Snapshot Highlighting Missing Values

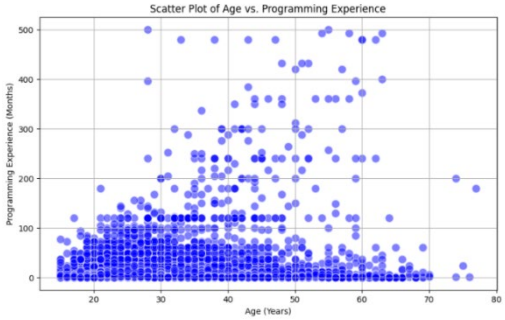
Evaluating response rates within our dataset is essential for gauging its completeness and directly impacts our approach to data preparation. Figure 3 elucidates the response rates for each attribute, revealing considerable variance. For instance, 'EmploymentStatus' and 'InterestInDevCareer' exhibit commendably high response rates, at 97.5% and 96.8% respectively, which suggests a high level of data reliability for these attributes. On the other hand, the 'AnnualIncomeUSD' attribute displays a notably lower response rate at 56%, indicating a significant portion of missing data. Such a low response rate for a crucial attribute necessitates a rigorous evaluation of potential data imputation methods or the consideration of exclusion of incomplete records. This evaluation must balance the benefits of a comprehensive dataset against the integrity of the analyses to be conducted, especially given the project's focus on the accuracy of income prediction for developers. In light of this, the strategy for handling these missing values will be pivotal, as it must align with the goal of minimizing bias while maximizing the utility of the remaining data.

[5]:

	Attribute	Missing Values	Response Rate (%)
WeeklyHoursSpent	WeeklyHoursSpent	1523	91.6
MonthsProgExp	MonthsProgExp	1431	92.1
EduLevel	EduLevel	921	94.9
EmploymentStatus	EmploymentStatus	453	97.5
AnnualIncomeUSD	AnnualIncomeUSD	7979	56.0
Age	Age	1023	94.4
InterestInDevCareer	InterestInDevCareer	589	96.8
Region	Region	892	95.1

Figure 3 – Table of Attributes and how many missing values they have

In addressing outliers in 'WeeklyHoursSpent', entries over 100 hours weekly were deemed unrealistic and excluded. The mean, set at 13 hours from credible responses, filled missing values, ensuring data integrity for analysis (Figure 4). For 'MonthsProgExp', we removed data suggesting programming experience over 75% of a respondent's lifetime or beyond 500 months, to correct data inaccuracies. This filtration, depicted in Figure 5, ensures our dataset's credibility, showing a realistic distribution of programming experience.



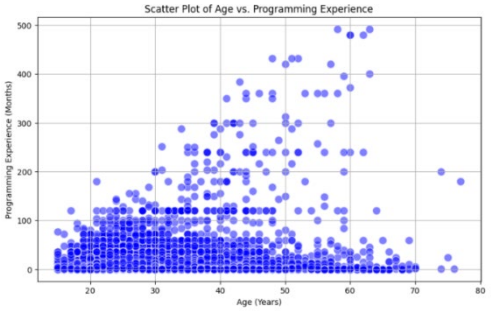


Figure 4 and 5 – Scatter plot of Age vs Experience before and after cleansing

In data cleansing, missing values in 'EduLevel' were scrutinized against regions to check for randomness. The uniform distribution of missing data, depicted in the bar graph (Figure 6), suggested randomness. Therefore, it was reasonable to categorize missing 'EduLevel' data as 'No Formal Education,' resulting in a new group of 921 individuals. This step ensured a more inclusive dataset for subsequent analyses.

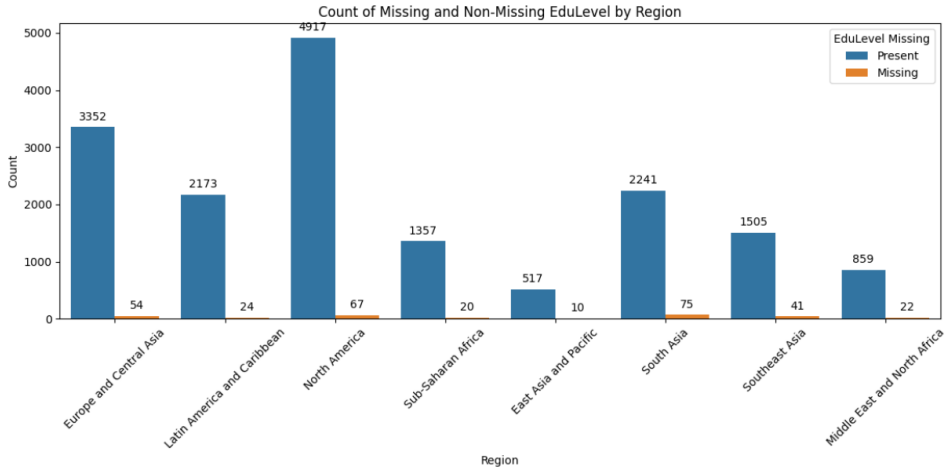


Figure 6 – Missing Values in each region

For the 'EmploymentStatus' attribute, we opted to remove the few missing entries due to a high response rate of 97.5%. This decision ensures the dataset's integrity with minimal impact on analysis, considering the random nature of the missingness. While data exclusion slightly reduces the sample size, it avoids potential biases from imputation. This trade-off is acceptable for maintaining the quality of our analysis.

In cleaning the 'Age' attribute, we aimed to remove unrealistic entries by excluding all ages under 16 and over 80, which are considered outside the typical working age range and likely to be input errors. This step ensures the robustness of our dataset by retaining only plausible age values for our analysis. After applying these filters, we addressed the remaining missing values by median imputation, setting them to 25, which is the median age of our respondents. This approach maintains the central tendency of our data without the influence of extreme values.

For the 'InterestInDevCareer' attribute with a 97% response rate, the small fraction of missing values (3%) was excluded to maintain analysis integrity, focusing only on entries with confirmed interest. This decision, assuming non-responses are randomly distributed and uncorrelated with other significant attributes, reduces potential biases from imputation. Thus, the dataset was refined to include only respondents with clear development career interests.

For the 'Region' attribute, crucial for income prediction, entries lacking regional data were excluded. This decision stemmed from the randomness of missing data, lacking any pattern hinting at collection bias or specific respondent traits. Given the attribute's high response rate and the dataset's robustness, this exclusion minimizes potential bias, prioritizing data integrity and accurate predictive modeling over speculative imputation for a key variable.

To address missing 'AnnualIncomeUSD' data, which showed no clear pattern against variables like 'Education Level' and 'Region', and faced with a 56% response rate, we opted to remove ambiguous responses and null values. Despite considering methods like predictive modeling and imputation, we chose removal to preserve data integrity and enhance model reliability. This step, while reducing dataset size, ensures our analysis is grounded in complete, accurate data, prioritizing quality over quantity for more dependable outcomes.

Transforming the 'AnnualIncomeUSD' categorical data into ordinal numerical levels streamlines our analysis, enabling us to treat income ranges as a spectrum rather than discrete, unrelated groups. This approach aligns with the inherent order of income levels, preserving the hierarchy from lowest to highest while simplifying the data's complexity.

Figure 7 exhibits a count plot, which visualizes the frequency distribution of the respondents across the transformed income levels. This visualization aids in quickly identifying the concentration of respondents within specific income brackets, facilitating an understanding of the socioeconomic distribution within the data. The count plot reveals that the majority of respondents fall within certain income brackets, indicating common income ranges in the dataset.

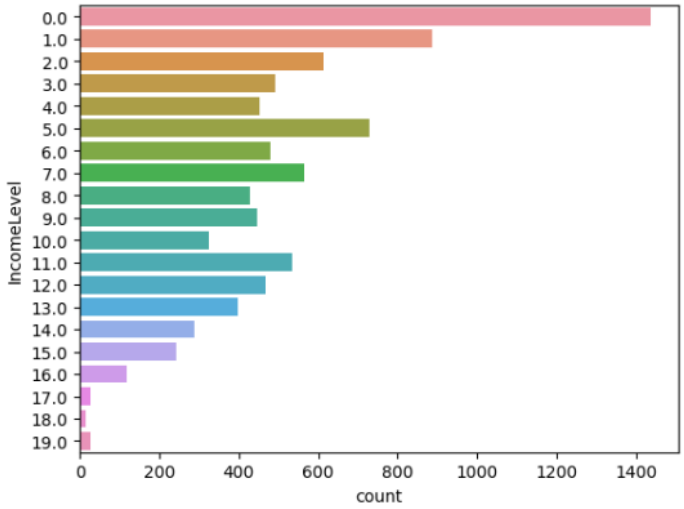


Figure 7- Ordinal Income Level Count Plot

By assigning an ordered numerical value to each income bracket, we can utilize various analytical and statistical methods that necessitate numerical input. This method not only enhances the clarity of our data but also maintains the meaningful categorization of income ranges, ensuring that our analysis respects the natural order of the income variable.

Such a transformation is particularly useful for revealing patterns and correlations that might not be apparent when dealing with purely categorical data. It provides a structured format for our dataset, making it amenable to advanced analytics.

For the updated DataFrame displayed in (Figure 8), we have conducted meticulous data cleaning to ensure that our dataset is primed for an effective exploratory data analysis (EDA). By removing unrealistic entries and addressing missing values with appropriate methods, we've streamlined the data to reflect only viable and realistic information. This refinement allows us to trust that the subsequent analysis is based on quality data, increasing the likelihood of uncovering genuine insights. The clean data lays a solid foundation for EDA, which can include various methods such as plotting distributions, identifying outliers, and exploring relationships between features to derive meaningful patterns and trends.

	WeeklyHoursSpent	MonthsProgExp	EduLevel	EmploymentStatus	AnnualIncomeUSD	Age	InterestInDevCareer	Region
1	10	6	Bachelor's degree	No	Under $1,000	38	I am already a developer	Latin America and Caribbean
4	2	24	Bachelor's degree	Yes	40,000to49,999	35	I am already a developer	East Asia and Pacific
5	10	50	Bachelor's degree	Yes	75,000to89,999	27	I am already a developer	North America
6	5	36	Some college credit, no degree	Yes	60,000to74,999	24	I am already a developer	North America
7	20	30	Bachelor's degree	Yes	1,000to2,999	23	I am already a developer	Europe and Central Asia
...	...	...	...	...	...	...	...	...
18119	30	15	Some college credit, no degree	No	Under $1,000	32	Yes	Europe and Central Asia
18120	10	1	Bachelor's degree	No	10,000to14,999	37	Yes	Europe and Central Asia
18123	10	13	Associate's degree	No	20,000to24,999	23	Yes	North America
18124	6	36	Professional degree (MBA, MD, JD, etc.)	Yes	30,000to34,999	34	I am already a developer	Europe and Central Asia
18125	5	200	Master's degree (non-professional)	No	10,000to14,999	44	No	Europe and Central Asia

8975 rows × 9 columns

Figure 8 – Showcases the cleansed Data Frame

Exploratory Data Analysis (EDA) of Important Attributes

WeeklyHoursSpent: Within the dataset, 'WeeklyHoursSpent' showcases the weekly amount of time participants engage in programming. From the pool of 8975 respondents, the average reported time is around 12 hours per week. This average suggests that the majority of participants spend a moderate amount of time coding, which might reflect either hobbyist coders or part-time professionals. The standard deviation, a substantial 12.3 hours, reveals that there's a broad range in the time commitment amongst participants. The maximum value observed in the dataset is 100 hours per week, indicating a reasonable upper limit for the most dedicated or possibly professional coders. At the median of 8 hours, we find that at least half of the respondents are likely balancing coding with other life activities.

MonthsProgExp: Analysis of the 'MonthsProgExp' attribute shows an average of about 15 months of programming experience, with a very wide range of responses as seen in the standard deviation of roughly 35.5 months. The data has been curated to exclude exceedingly high reports of experience that surpass 500 months, which were considered to be unrealistic. This curation aims to enhance the representativeness of the dataset by focusing on credible and verifiable ranges of experience. The median of 3 months suggests that the majority of the participants are relatively new to programming.

Age: The age attribute of the dataset presents an average of approximately 30 years with a standard deviation close to 9.4 years. This indicates a relatively young participant base, which is further corroborated by half of the respondents being under the age of 28. Extreme age outliers have been

addressed by restricting the dataset to include individuals between the ages of 15 and 80, offering a more focused view on the target demographic likely to be engaging in programming.

Overall Insights: The centralized measures and tailored data cleaning steps undertaken for attributes like 'WeeklyHoursSpent' and 'MonthsProgExp' underscore our commitment to maintaining data integrity and reliability. By filtering out implausible entries, we ensure that subsequent data analysis and modeling efforts are grounded in verifiable data. This thoughtful curation facilitates a more accurate and nuanced understanding of the coding practices, experience levels, and demographics within the dataset. The refinement of the data, with attention to detail and context, sets a strong foundation for insightful exploratory and predictive analysis.

	WeeklyHoursSpent	MonthsProgExp	Age
count	8975.000000	8975.000000	8975.000000
mean	11.904401	14.800780	30.470529
std	12.272389	35.484067	9.360223
min	0.000000	0.000000	15.000000
25%	4.000000	1.000000	24.000000
50%	8.000000	3.000000	28.000000
75%	15.000000	12.000000	35.000000
max	100.000000	492.000000	77.000000

Figure 9 – Snapshot of Descriptive Analysis of Numerical Attributes

From the pie chart in figure 10, we can infer that the most common educational level among respondents is a Bachelor's degree, making up nearly 38% of the sample. This suggests that the majority of the individuals in this group have completed undergraduate education, which might be reflective of a standard requirement or a common benchmark in their professional fields.

The second most reported education level is some college credit with no degree, followed by a Master's degree (non-professional). These categories suggest a significant portion of respondents have pursued higher education but may not have completed it or have advanced to a graduate level of study. The presence of respondents with professional degrees and Ph.D. indicates that there is also a representation of highly educated professionals within the sample.

The smaller segments, such as 'No high school (secondary school)' and 'No Formal Education,' while being a minor proportion, highlight the diversity in educational backgrounds. These might represent outliers in the dataset or could indicate individuals in roles where formal education is not as heavily weighted for employment.

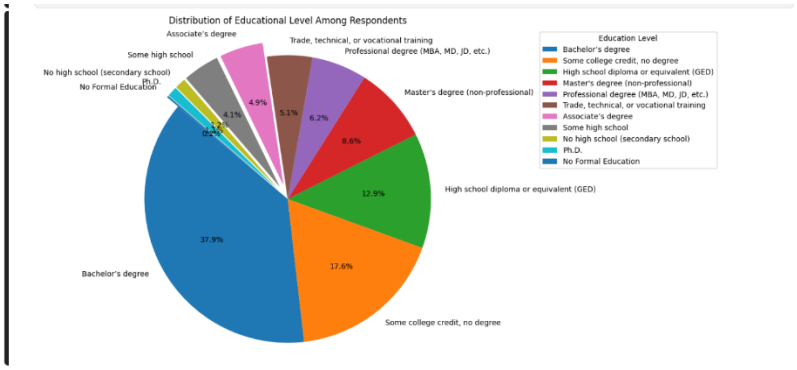


Figure 10 – Pie chart of Distribution of Educational Level Among Respondents

The distribution shown in the pie chart (figure 10) offers valuable insights into the educational diversity within the dataset and sets a foundation for further analysis on how education level might correlate with other variables such as income, employment status, or programming experience.

Figure 11, "Mean MonthsProgExp Across Income Levels," shows a bar chart of average programming experience by income. Generally, higher income correlates with more experience. Notably, there's a significant increase at the second-highest income level, possibly pointing to seasoned experts. The decrease at the highest income level suggests a cap on income with increased experience or factors outside this dataset, like industry shifts or senior roles less focused on programming.

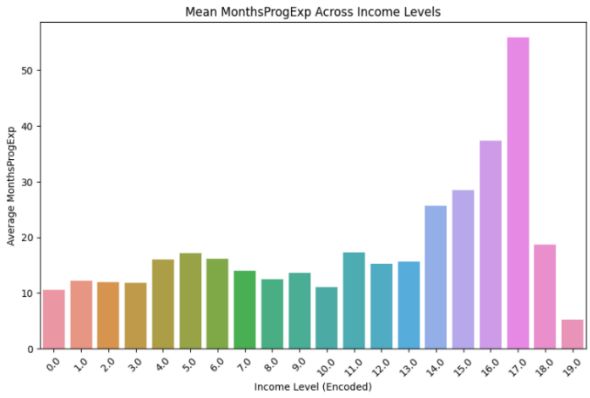


Figure 11- Average experience over ordinal income levels

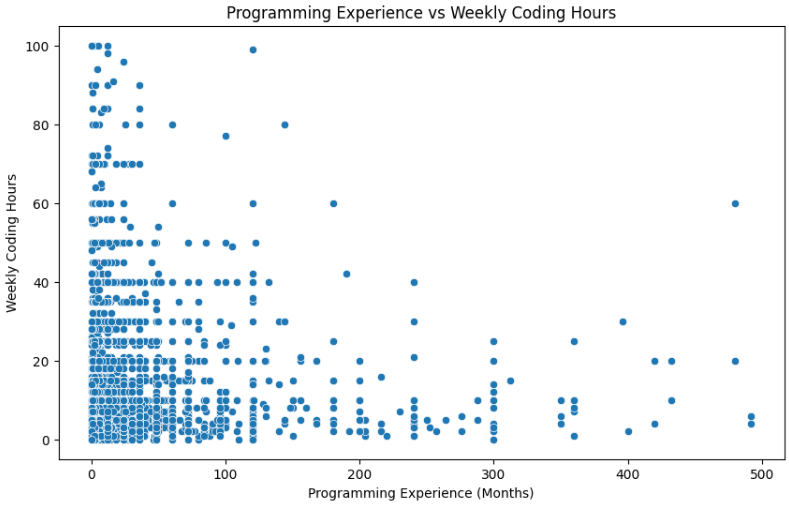


Figure 12- Scatter plot of Weekly coding hours with monthly experience

Figure 12 shows many respondents are newcomers to programming, with 0 to 100 months of experience, displaying varied weekly coding hours. With increased experience, weekly coding hours become more variable, highlighting the diminished direct correlation between experience and coding hours for seasoned programmers. This suggests factors like job roles, personal commitments, or shifts to managerial positions might influence coding time. Outliers include mid-experience coders with high weekly hours and highly experienced coders with low weekly coding, reflecting the survey participants' diverse circumstances.

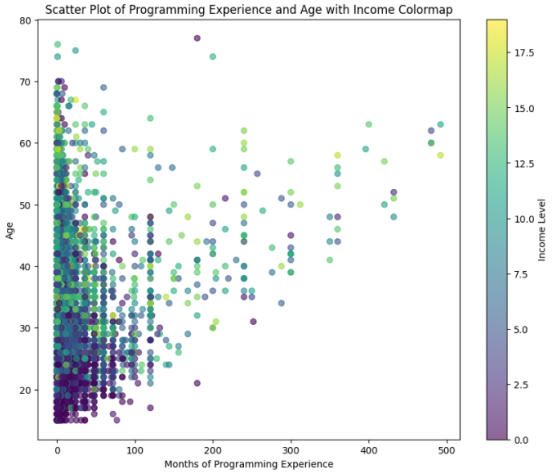


Figure 13- Scatter plot showing us relationships between age and experience

Figure 13's scatter plot illustrates the complex link in the computer industry between age, programming experience, and income. It emphasizes that years of experience in the industry or advanced age may not always translate into more income, pointing to the impact of specialized training, consumer demand, or regional economic variables on wages. The data illustrates the vast appeal and variety of entry points of the area of programming, with a youthful influx occurring alongside extensive age representation. The disparity in pay between experience levels defies the notion that greater experience equates to better salary, highlighting the distinctive nature of programming careers and the myriad of variables influencing their path. This emphasizes how dynamic the tech sector is and how factors other than years of experience can have an impact on a person's professional success.

Section 3 – Cluster Analysis

Clustering is a data mining method used to group similar data points together. It's useful for discovering hidden patterns in a dataset, such as in the coding community's behaviors and characteristics. In our analysis:

1. We used cluster analysis to uncover natural groupings based on programming experience, education, employment, etc.
2. Categorical variables like education levels were numerically encoded for the clustering algorithms.
3. Data normalization was carried out using StandardScaler to ensure equitable feature influence during clustering.
4. The k-means algorithm was applied to partition the data. The key parameters were:
    - n_clusters = 3, chosen by the elbow method to determine the number of clusters without undue complexity.
    - Init = 'k-means++' for optimal centroid placement.

- max_iter = 300 and n_init = 10, ensuring thorough algorithm runs and convergence on a solution.
- random_state = 0 for consistent results across runs.

5. **High Income Group**: As shown in figure (14) The WCSS decreases sharply as the number of clusters increases from 1 to around 3 and then slowly tapers off. This suggests that beyond 3 clusters, adding more clusters doesn't significantly improve the model. Hence, for the high-income group, 3 clusters might be the optimal number as the elbow seems to be at that point.
6. **Low Income Group**: In figure (15) Similar to the high-income group, the WCSS for the low-income group decreases sharply up to k=3 and then plateaus. This indicates that for the low-income group, the optimal number of clusters also appears to be 3, as adding more clusters beyond this point leads to diminishing returns in terms of decreasing WCSS.

In both cases as shown in figure 14 and 15, the elbow method graph indicates that k=3 is a good choice for the number of clusters. This makes sense because it represents a point where adding more clusters doesn't provide as much benefit in terms of variance explained. In other words, k=3 provides a reasonable trade-off between the complexity of the model and the similarity of instances within the clusters.

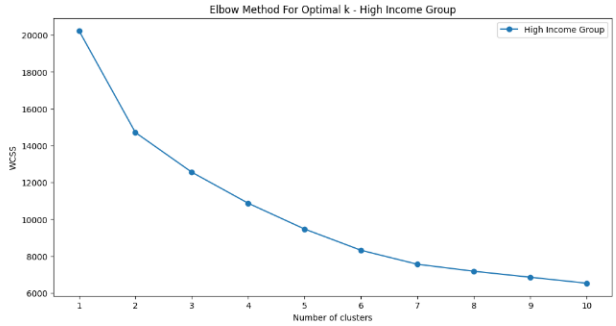


Figure 14- Optimal k for high income group

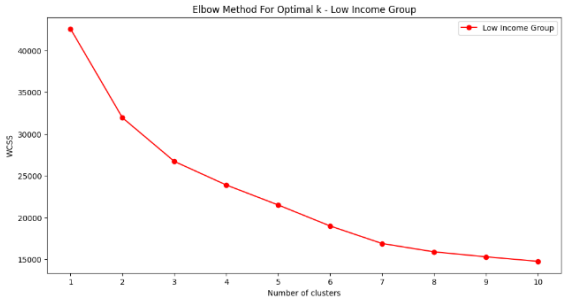


Figure 15- Optimal k for low income group

High-Income Subset Cluster Analysis

From the plot in figure 16, we can infer the following:

Cluster 0 suggests novices with minimal weekly coding hours, possibly coding as a secondary activity or just starting.

Cluster 1 shows a mixed group with varied experience and learning hours, indicating a blend of steady learners and intermittent coders.

Cluster 2 represents seasoned coders with a wide experience range, some of whom continue to invest significant time in learning, reflecting either high commitment or a need to update skills frequently.

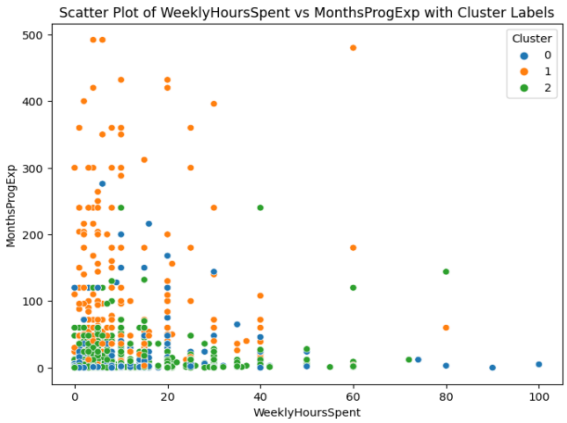


Figure 16- Clusters of Experience vs Weekly hours spent coding

The distribution suggests that there might be a trend where individuals who spend more hours coding each week tend to have more months of programming experience, which is particularly visible in Cluster 2. However, there is considerable overlap between clusters, especially between Clusters 0 and 1, indicating that weekly coding hours do not strictly predict the level of programming experience.

Furthermore, there are outliers, particularly in Cluster 2, where individuals have a high number of programming months but varying weekly coding hours, including some who code very little each week. These outliers might represent experienced programmers who have moved into managerial or less hands-on technical roles.

From the plot in figure 17, the following observations can be made:

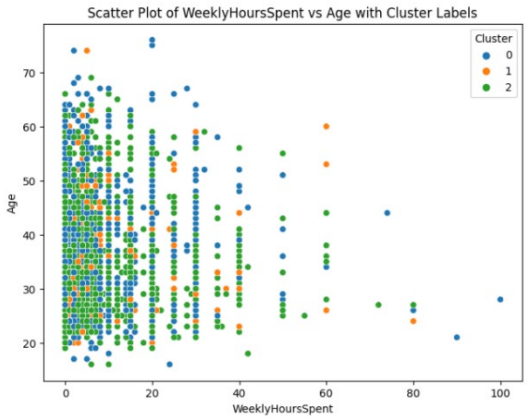


Figure 17- Cluster Scatter plot of Weekly hours spent coding vs age

Cluster 0 includes a broad age range with few weekly coding hours, hinting at casual coding across all ages.

Cluster 1's diverse age spread and variable coding hours indicate a mix of individuals, from seasoned professionals updating their skills to late bloomers intensifying their learning.

Cluster 2 consists of dedicated coders of all ages, potentially including career builders and those in rigorous training phases. Age doesn't strictly predict coding hours; however, a trend suggests older individuals may code less weekly, with a few younger outliers engaging extensively. These insights reflect the varying commitment to coding within the high-income segment.

The distribution suggests no clear trend linking age with weekly coding hours across clusters, showcasing a broad age range within each. However, a tendency for older individuals to code less frequently is observed, with few exceptions of younger individuals engaging in extensive coding hours weekly. Outliers, especially in Cluster 2, highlight varied coding engagements that defy age norms, offering a glimpse into diverse coding practices among different age groups in the high-income segment.

In the provided bar graph in figure 18, each cluster represents a distinct grouping of individuals based on their educational levels, which can be indicative of their professional background and potential roles within the workforce.

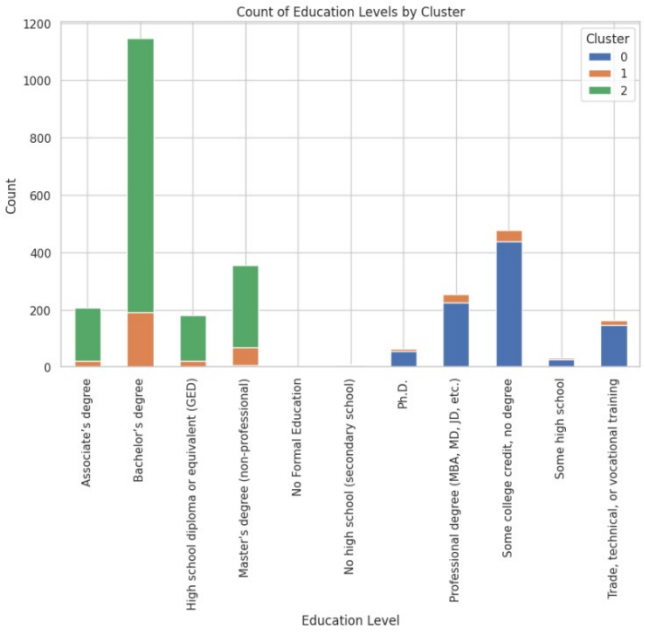


Figure 18- Bar graph of the count of education levels with clusters

Cluster 0's educational diversity, notably lacking Bachelor's degree holders, hints at professions where technical training or professional degrees suffice. Cluster 1's varied educational levels suggest it includes individuals in transitional career phases, possibly blending early to mid-career professionals. Cluster 2, rich in Bachelor's and Master's degree holders, likely represents the workforce's highly educated sector, involved in specialized or academic fields. This overview highlights educational diversity's role in understanding workforce trends and informs strategies in education and recruitment.

Overall, the bar graph illustrates the diversity in educational achievements across the clusters, which can be leveraged to draw conclusions about the professional landscape of the dataset's respondents. It also underscores the importance of considering educational background when analyzing labor market trends, recruitment strategies, or policy-making in education and workforce development.

Low-Income Subset Cluster Analysis

Cluster 0: This cluster predominantly consists of individuals with lower months of programming experience (largely under 100 months) and generally spending fewer hours per week coding. This might represent a group with either newer entrants into the programming field or those who program part-time or as a hobby.

Cluster 1: This group shows a spread of experience from low to high (0 to nearly 500 months), but like Cluster 0, they also spend fewer weekly hours coding. The wide range of experience suggests this cluster could include both seasoned programmers who now code less frequently and beginners who are the same.

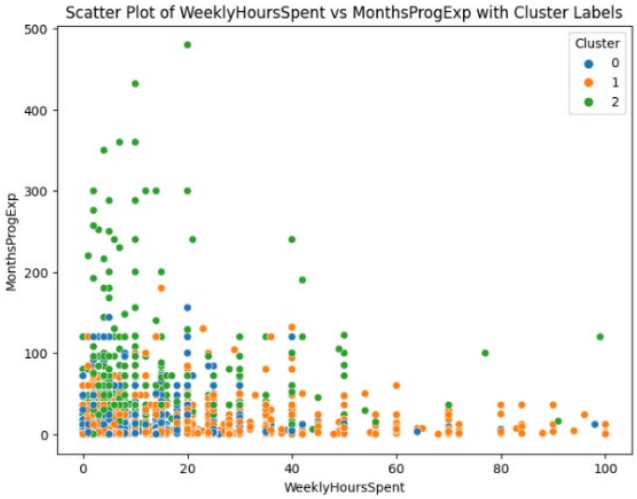


Figure 19- Clusters of Experience vs Weekly hours spent coding

Cluster 2: This cluster stands out with a smaller number of individuals who have a high number of programming experience months (exceeding 100 months and going up to 500), with a wider range of weekly coding hours. This could represent highly experienced programmers who vary in their current coding activity, possibly those in senior or specialized positions that do not require extensive coding hours each week.

The plot indicates a potential trend where, regardless of the cluster, as programming experience increases, the number of weekly hours spent coding does not necessarily increase. This could be due to several factors such as shifts in career to management or consultancy roles where hands-on coding decreases despite the high level of experience.

Figure 20's scatter plot depicts programming experience against age, revealing three clusters:

Cluster 0: Younger individuals (20-30 years), often new to coding with less than 100 months of experience, suggesting they are early in their coding careers.

Cluster 1: A wide age range (20-60 years) with experience up to 200 months, indicating a mix of mid-career coders and those shifting into coding from other fields.

Cluster 2: Older individuals, mostly over 30, with diverse experience levels, including some with over 400 months of experience, likely indicating long-term industry veterans or seniors. The data suggests age isn't directly proportional to coding experience, highlighting the varied entry points and career progressions in tech.

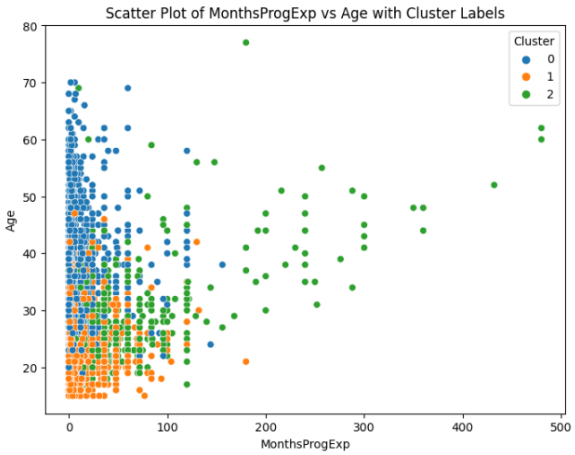


Figure 20- Cluster Scatter plot of Weekly hours spent coding vs age

Overall, the plot indicates that there isn't a strong correlation between age and months of programming experience across the clusters. Younger individuals tend to have less experience as expected, but beyond a certain point, the amount of experience doesn't increase consistently with age. This could reflect the diversity of pathways through which individuals engage with coding and software development careers.

In figure 21:

Cluster 0: Predominantly includes individuals with a bachelor's, followed by those with master's, associate's, and Ph.D. degrees, indicating a highly educated group, likely in professions requiring advanced degrees.

Cluster 1: Primarily consists of bachelor's degree holders and has the most individuals without any degree and a notable count with professional degrees, reflecting a diversity in educational backgrounds, from self-taught individuals to career professionals.

Cluster 2: similar to the other clusters, includes many individuals with a bachelor's degree but has comparatively fewer counts in other education levels.

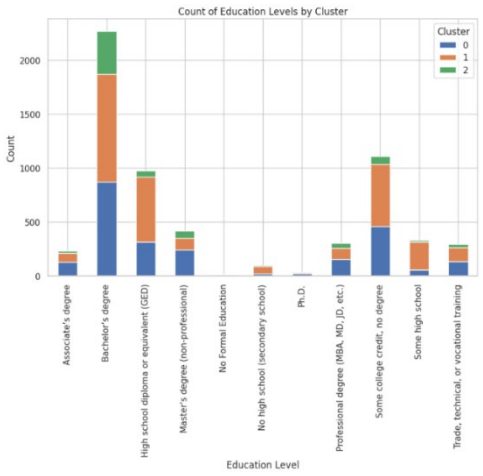


Figure 21- Bar graph of the count of education levels with clusters

The distribution reveals a bachelor's degree as the predominant educational qualification, indicating its role as a common industry baseline. The varied presence of professional degree holders and Ph.D.'s across clusters points to a field inclusive of specialized or research roles. Notably, cluster 1's substantial "no degree" demographic highlights the tech industry's openness to self-taught professionals and those with non-traditional education paths.

Cluster Analysis Conclusions

High-Income subset:

The cluster analysis of high-income individuals reveals:

Cluster 0: Aspiring enthusiasts marked by less experience and lower weekly coding hours, this group seems to include novices and those with non-traditional or evolving career paths, balancing coding with other responsibilities.

Cluster 1: Diverse professionals featuring a wide range of experience and learning hours, this group blends self-taught individuals, career switchers, students, and professionals maintaining industry relevance.

Cluster 2: Seasoned experts characterized by substantial experience and variable coding hours, this cluster consists of veterans, possibly in senior or less coding-intensive roles, and often with higher educational qualifications, correlating with high-income tech roles.

Low-Income Subset:

In the low-income bracket:

Cluster 0: Emerging Coders, this group is at the outset of their coding careers with minimal experience and weekly coding hours, including part-time coders and those with advanced degrees, potentially in career transition.

Cluster 1: Diverse Contingent, with varying experience and modest weekly coding, this cluster might include seasoned programmers in less coding-intensive roles and self-taught professionals, highlighting coding's accessibility beyond formal education.

Cluster 2: Seasoned Part-Timers, characterized by significant experience yet varied coding hours, members may be established coders with less need for frequent practice, possibly due to advanced positions or expertise.

Section 4 - Machine Learning for Classification and their Implementation:

Machine learning is a branch of artificial intelligence that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. It enables computers to perform specific tasks from data without explicit programming. Among various types of tasks that machine learning models perform, classification is one of the most important.

Classification in machine learning is a supervised learning approach where the model is trained on labeled data. The training involves providing the model with input-output pairs, where the inputs are features or attributes, and the outputs are categories or classes. The goal is for the model to learn the relationship between the attributes and the classes so that it can predict the class of new, unseen instances based on their attributes.

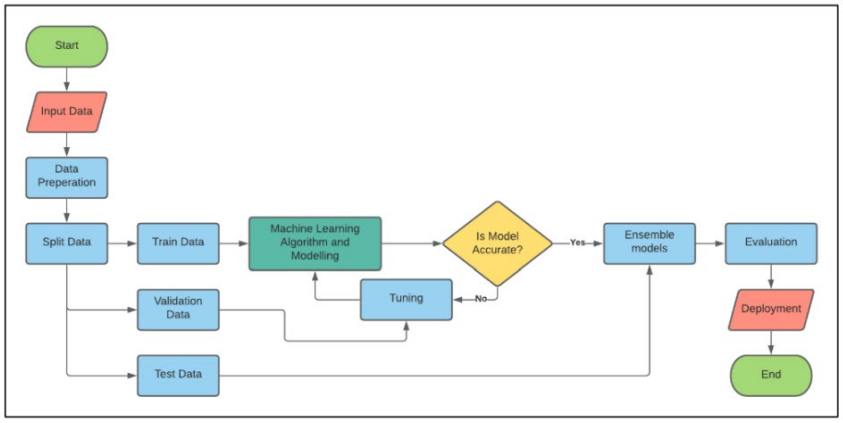


Figure 22- Machine learning workflow

The significance of classification in data analysis and machine learning cannot be overstated. It is used in a wide array of applications from email filtering (spam or not spam), image recognition (identifying objects within images), to medical diagnosis (classifying tumors as benign or malignant). Classification helps in making informed decisions based on data patterns and is essential in areas where the rapid and accurate categorization of items is critical. It's a powerful tool for pattern recognition, decision making, and prediction, making it indispensable in both commercial and research settings.

Refer to the attached flow-chart in figure 22 as a visual representation of the machine learning workflow. This chart guides the process from initial data input to the deployment of a predictive model.

Data Preparation

The first step, data preparation, involves cleaning the dataset, dealing with missing values, and converting categorical data into a machine-readable format using encoding techniques. This ensures that the dataset is ready for effective model training.

Data Splitting

Data is divided into training, validation, and test subsets. The training set is used to teach the model to recognize patterns, the validation set helps in fine-tuning model parameters, and the test set provides an unbiased evaluation of the final model fit.

Training and Validation

The training phase involves algorithms learning from the data. The validation phase is crucial for selecting the best-performing model and involves techniques like cross-validation, which helps ensure the model's generalizability.

Machine Learning Algorithms for Modeling

Three classification algorithms are selected for this task: k-Nearest Neighbors, Decision Trees, and Logistic Regression. Each method offers unique benefits, such as k-NN's non-parametric nature, the interpretability of decision trees, and logistic regression's efficacy in binary classification problems.

Hyperparameter Setting and Tuning

Hyperparameter tuning is a critical step to improve model performance. Techniques like grid search and random search, accompanied by cross-validation, help in identifying the optimal settings for each classification method.

Ensemble Models

When appropriate, ensemble methods like bagging or boosting may be employed to enhance predictions. These methods combine several base models to produce one optimal predictive model, often resulting in improved accuracy and stability.

Model Evaluation

Model accuracy is gauged through metrics like precision, recall, and the F1 score. The model that best predicts the high-income developers based on their information is chosen for deployment.

Data Transformation and Normalization

Transformation and normalization standardize the range of independent variables, ensuring that the model is not skewed by the scale of any feature. This homogenization of data improves algorithm

performance and is especially beneficial for algorithms like k-NN, which rely on distance calculations.

Conclusion

In summary, the proposed machine learning classification approach is methodically designed to predict developers' income levels with high accuracy. The selected classification methods are apt for the nature of the data, and the thorough process of data preparation, model training, tuning, and evaluation is tailored to yield a robust predictive model.

Chosen Models:

k-Nearest Neighbors Model

The k-Nearest Neighbors (k-NN) algorithm, known for its simplicity, is chosen for predicting high-income brackets among developers due to its effective use of the proximity principle. It's particularly suitable for our dataset's complex decision boundaries due to its non-parametric nature and intuitive voting mechanism. Initially, k-NN achieved a 76.6% accuracy, indicating its capability to differentiate income classes. After optimizing parameters through a grid search, its accuracy improved to 78.2%, highlighting the importance of hyperparameter tuning. This process not only increased accuracy but also enhanced precision and recall, demonstrating k-NN's potential in high-income prediction tasks.

Random Forest Classifier

The Random Forest (RFC) algorithm is celebrated for its robustness in classification, leveraging numerous decision trees to improve accuracy and prevent overfitting. This method excels in handling complex datasets and is chosen for its ability to discern high-income brackets among developers through its ensemble approach and feature importance evaluation. Starting with a base accuracy of 76.4%, RFC demonstrated promising initial results. Through hyperparameter tuning, including adjustments to tree counts and depth, we improved its performance to 79.4%. This optimization process showcased RFC's flexibility and effectiveness in capturing the dataset's nuances, confirming its utility in identifying high-income profiles with increased precision and recall.

Gradient Boosting Machines (GBM) or XGBoost

XGBoost is celebrated for its classification efficacy, using a gradient boosting framework to handle complex data. Its speed, accuracy, and tunable parameters make it ideal for predicting high-income brackets. Initially, XGBoost demonstrated a promising 78% accuracy, highlighting its capability to differentiate income levels effectively. A randomized grid search refined the model, optimizing hyperparameters to boost accuracy to 80.04%, proving the importance of hyperparameter tuning in enhancing predictive accuracy. This process emphasized the critical role of careful feature selection and parameter optimization, achieving significant improvements and demonstrating XGBoost's strength in predicting high-income status with refined parameters like 'subsample': 1.0, 'reg_lambda': 0.5, and 'learning_rate': 0.1 among others.

Neural Network

The Neural Network Model for predicting high-income brackets showcases the efficacy of deep learning for complex data. Starting with a basic architecture, the base model achieved an accuracy of 79.67%, indicating its ability to discern patterns within the data. Through hyperparameter optimization via a randomized search, the model's accuracy improved to 81%, reflecting a refined ability to generalize and predict accurately.

This optimization process highlights the critical role of tuning in enhancing model performance, with the adjusted hyperparameters significantly boosting accuracy. The success of the optimized Neural Network underscores the power of deep learning and strategic hyperparameter adjustments in improving predictions in classification tasks. The key parameters leading to this improvement are detailed in the study, showcasing the tailored approach to maximizing the model's predictive capabilities.

Ensemble Methods

Random Forest Classifier with Support Vector Machine (RFC with SVM)

In the quest to refine the predictive capabilities of our models, we explore the ensemble technique that synergizes the strengths of both Random Forest Classifier (RFC) and Support Vector Machine (SVM). This ensemble method aims to leverage the diverse decision-making approaches of RFC's ensemble of decision trees with SVM's margin-based classification, providing a balanced and robust predictive model.

Ensemble Justification and Methodology: The Random Forest algorithm, known for its high accuracy, operates by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes of the individual trees. Support Vector Machine, on the other hand, is a powerful classifier that works well on a clear margin of separation and is effective in high-dimensional spaces. By combining these two models, we aim to capture a broader spectrum of the data's underlying patterns, thereby enhancing the final prediction accuracy.

The ensemble was created using a two-step process:

1. Hyperparameter Optimization: We performed a grid search for both RFC and SVM to determine the best individual parameters for each model. The optimized parameters aim to tailor each model closely to our dataset's specific characteristics.
2. Model Retraining and Evaluation: With the best parameters identified, we retrained both models and then evaluated their combined accuracy through a voting mechanism. This technique takes into account the predictions from both models, seeking consensus to finalize the classification.

Results and Performance: Upon evaluation, the ensemble model achieved an accuracy of 79.61%, which surpasses the individual accuracy of the Random Forest Classifier (79.39%). Although the improvement is modest, it reflects the potential of combining different modeling techniques to create a more accurate predictive system.

Random Forest Classifier with Gradient Boosting Machines and Soft voting (RFC with GBM)

In our continuous effort to enhance the predictive strength of our models, we delve into the ensemble approach that merges the robust Random Forest Classifier (RFC) with the efficient Gradient Boosting Machines (GBM). This combination utilizes the collective capabilities of both models, with RFC's bagging technique and GBM's boosting strategy, aiming to produce a superior composite classifier.

Ensemble Justification and Methodology: The Random Forest classifier brings its ability to reduce variance by training on different subsets of the data and using averaging to improve accuracy. GBM, distinguished for sequentially correcting the mistakes of weak learners and focusing on the difficult parts of the dataset, provides a complementary approach. Together, they form a potent alliance for handling diverse data patterns and complexities.

The ensemble method proceeded in the following stages:

1. Hyperparameter Optimization: We meticulously tuned both RFC and GBM using their respective grid searches, fine-tuning the models to our dataset's nuances and extracting the most potent parameter set for each.
2. Soft Voting Mechanism: Equipped with the best hyperparameters, we employed a soft voting classifier that considers the probability estimates from RFC and GBM. This method emphasizes consensus building, giving weight to the confidence level of each model's predictions.

Results and Performance: The synergized model, upon execution, displayed an impressive accuracy of 80.06%, edging past the individual performance of RFC (79.39%) and closely matching GBM on its own (80.39%). This increment, while seemingly slight, is a testament to the synergy achieved by integrating different machine learning paradigms, validating the premise that ensemble methods can unlock new levels of accuracy and reliability in predictive modeling.

Results

Model Type	Accuracy (to 2d.p)
k-Nearest Neighbors	78.22%
Random Forest Classifier (RFC)	79.39%
Gradient Boosting Machines (GBM)	80.39%
Neural Network	**81%**
RFC with SVM	79.61%
RFC and GBM with Soft Voting	80.06%

Figure 23- Table of models and their accuracy

Section 5 – Evaluation of Machine Learning Models

In evaluating the performance of our machine learning models, we observe from the results table a hierarchy in accuracy scores that offers insights into each model's effectiveness. The k-Nearest Neighbors (k-NN) model provided a foundational accuracy of 78.22%, establishing a benchmark for

comparison. Progressing to more complex models, the Random Forest Classifier (RFC) improved upon this with a score of 79.39%. A notable jump in performance was seen with the Gradient Boosting Machines (GBM), which achieved an accuracy of 80.39%, demonstrating the power of ensemble learning techniques.

Introducing neural network architectures led to a slight advancement, with an accuracy of 81%, underscoring the potential of deep learning models in capturing complex patterns within the data. Meanwhile, an ensemble of RFC with SVM yielded a combined accuracy of 79.61%, suggesting a synergistic effect, albeit modest. A soft voting ensemble combining RFC and GBM further harnessed the strengths of both models to reach an accuracy of 80.06%, illustrating the benefit of aggregating predictions.

To further understand and validate the performance of these models beyond accuracy, we will delve into confusion matrices and ROC curves. A confusion matrix provides a detailed breakdown of a model's predictions, showcasing the number of true positives, false positives, true negatives, and false negatives. This matrix is crucial for evaluating the precision and recall of a model, especially in datasets with class imbalances.

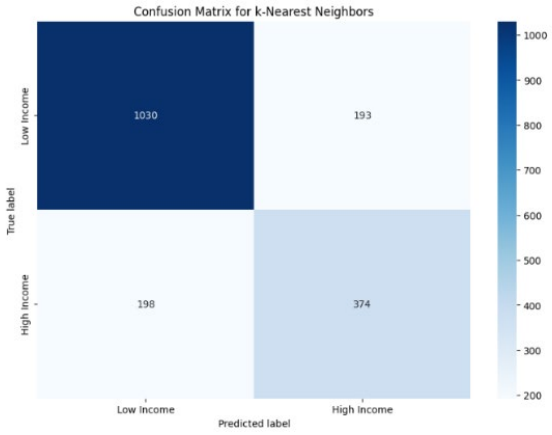
ROC curves, or Receiver Operating Characteristic curves, plot the true positive rate against the false positive rate at various threshold settings. The area under the ROC curve (AUC) is a measure of the model's ability to discriminate between the classes. A higher AUC indicates better model performance, with a value of 1.0 representing a perfect model and 0.5 a model with no discriminative power.

By utilizing these evaluation tools, we can paint a more comprehensive picture of our models' performance, taking into account not just the accuracy but also how well the models manage different types of classification errors.

Confusion Matrices

K-NN Confusion Matrix:

From the provided confusion matrix for the k-Nearest Neighbors classifier, we can interpret the following:



- **True Negative (TN): 1030** - The model correctly predicted a large number of individuals as Low Income. This indicates the model's ability to identify the negative class.
- **False Positive (FP): 193** - These are instances where the model incorrectly predicted High Income for individuals who are actually in the Low-Income category. The lower number here suggests a lower rate of this type of error.
- **False Negative (FN): 198** - Here, the model predicted Low Income when the individuals were High Income. The relatively low number

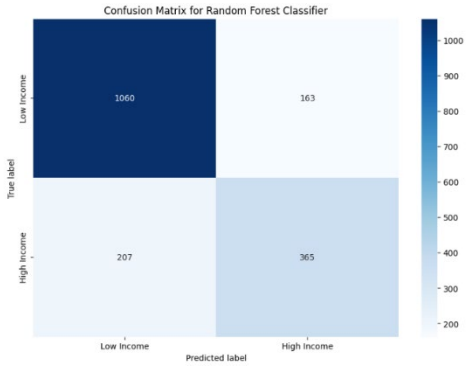
suggests that the model is quite good at catching the High-Income cases but there's still room for improvement.

- **True Positive (TP): 374** - This shows the number of times the model correctly identified High Income individuals, which is significantly lower than the True Negatives.

The confusion matrix visualization indicates that while the model is relatively conservative, preferring to predict Low Income, it still maintains a balanced approach between the precision and recall for the High-Income class. The color gradient emphasizes that the majority of predictions fall into the True Negative category, suggesting that the model may be biased towards predicting the more prevalent class in the dataset.

RFC Confusion Matrix:

The confusion matrix visualized here represents the performance of the Random Forest Classifier on the task of predicting income levels. In the context of your classification task, where the positive class is 'High Income' and the negative class is 'Low Income', the matrix can be read as follows:



True Negatives (Top-Left Quadrant): The model correctly predicted 'Low Income' 1,060 times. This means that for 1,060 instances, the model's prediction of a lower income bracket was accurate.

False Positives (Top-Right Quadrant): The model incorrectly predicted 'High Income' 163 times when the true label was 'Low Income'. These are Type I errors, indicating instances where the model overestimated the income level.

False Negatives (Bottom-Left Quadrant): The model incorrectly predicted 'Low Income' 207 times when the true label was 'High Income'. These are Type II errors, showing underestimation of income level.

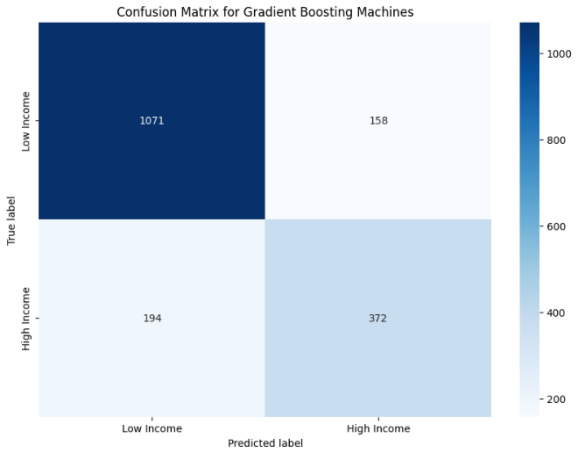
True Positives (Bottom-Right Quadrant): The model correctly predicted 'High Income' 365 times, which implies that it accurately identified the higher income bracket in these cases.

The matrix shows a relatively strong performance in predicting 'Low Income' correctly (True Negatives), while it shows room for improvement in correctly identifying 'High Income' (True Positives). The False Negatives indicate cases where the model might be overly conservative, marking potential high-income earners as low-income. Conversely, the lower number of False Positives suggests the model is less likely to incorrectly classify low-income earners as high-income.

GBM Confusion Matrix:

The confusion matrix depicted here illustrates the performance of the Gradient Boosting Machines (GBM) classifier. It outlines the counts of true and false predictions against the actual labels for a binary classification task, aimed at distinguishing between 'Low Income' and 'High Income' groups.

- **True Negatives (Top-Left Quadrant):** The model correctly predicted 'Low Income' for 1,071 individuals. This quadrant shows the instances where the GBM model accurately identified individuals in the lower income bracket.



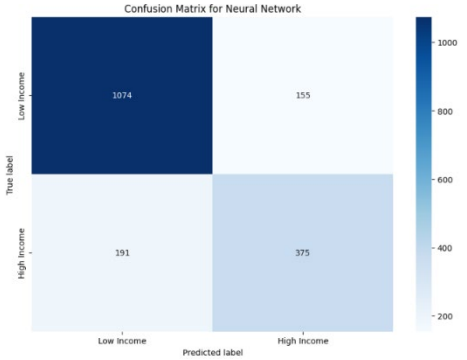
- **False Positives (Top-Right Quadrant):** There were 158 occurrences where the model incorrectly predicted 'High Income' for individuals who were in the 'Low Income' bracket. These are considered Type I errors.
- **False Negatives (Bottom-Left Quadrant):** The model predicted 'Low Income' for 194 individuals who were in the 'High Income' group. These mistakes represent Type II errors, where the model was overly conservative, failing to identify the correct higher income status.
- **True Positives (Bottom-Right Quadrant):** The model correctly identified 372 individuals as being in the 'High Income' group. This figure represents the GBM model's accuracy in predicting the higher income bracket.

This confusion matrix demonstrates that GBM is more adept at correctly identifying individuals with low income compared to high income, given the higher count of True Negatives to True Positives. The relatively low number of False Positives compared to False Negatives suggests that while the model is quite specific (few individuals are incorrectly labeled as high income), it could potentially be more sensitive (missing some individuals who do indeed have a high income).

Neural Network Confusion Matrix:

The confusion matrix for the Neural Network model presented here shows the model's performance in classifying individuals into 'Low Income' and 'High Income' groups.

- **True Negatives (Top-Left Quadrant):** There are 1,074 individuals correctly classified as 'Low Income'. This high number of true negatives indicates the model's strength in accurately identifying the lower income group.



- **False Positives (Top-Right Quadrant):** The model incorrectly predicted 'High Income' for 155 individuals who are in the 'Low Income' category. These errors may be less critical if the cost of false positives is low in the specific application context.
- **False Negatives (Bottom-Left Quadrant):** For 191 individuals, the model erroneously predicted 'Low Income' despite them being in the 'High Income' bracket. False negatives are

often seen as more problematic because they represent missed opportunities or undervalued potential in many applications.

- **True Positives (Bottom-Right Quadrant):** There are 375 individuals correctly classified as 'High Income'. The model has shown a good ability to identify higher-income earners but not as effectively as it identifies lower-income earners.

From the data in the confusion matrix, the Neural Network model demonstrates a balanced performance in detecting both income classes, with a slightly higher success rate in identifying low-income individuals compared to high-income ones. The comparable numbers of false positives and false negatives suggest that the model has a balanced type I and type II error rate.

ROC Curve

An ROC (Receiver Operating Characteristic) curve is a graphical representation that illustrates the diagnostic ability of a binary classifier system. It does so by plotting the True Positive Rate against the False Positive Rate at various threshold settings. The curve provides an aggregated measure of performance across all classification thresholds. One of the most important aspects of the ROC curve is the AUC value, or Area Under the Curve, which quantifies the overall ability of the model to discriminate between the positive and the negative class. An AUC of 0.5 suggests no discriminative power, akin to random guessing, while an AUC of 1.0 represents perfect discrimination.

Model	AUC (Area Under Curve)
k-Nearest Neighbors	83%
Random Forest Classifier	86%
Gradient Boosting Machines	86%
Neural Network	86%
RFC with SVM	85%
RFC and GBM with Soft Voting	86%

From the provided AUC scores for the different models, we can rank them in terms of their discriminative capabilities:

1. Gradient Boosting Machines (GBM), Neural Network, Random Forest Classifier (RFC), and Random Forest Classifier (RFC) with Gradient Boosting Machines (GBM) with Soft Voting: All share the highest AUC of 86%, indicating a strong ability to distinguish between low-income and high-income classifications.

2. RFC with SVM: With an AUC of 85%, this ensemble model demonstrates a robust performance, although it is not quite as high as the leading models.

3. k-Nearest Neighbors (kNN): It has the lowest AUC of 83%, which suggests it is less capable of discrimination compared to the other models.

Section 6 – Conclusion on Machine Learning Coursework

In this small study, we used a range of machine learning models to investigate the factors impacting the income levels of novice programmers. A multi-dimensional view of the aspects that can affect a coder's earning potential was supplied by the selected attributes, which included hours spent coding each week, months of programming experience, age, and encoded values of education level, region, and employment status.

The analysis showed that different traits had different predictive weights for different income levels. Some characteristics, such as months of programming experience and educational attainment, had a greater influence than others. A more sophisticated knowledge of how a person's income can be shaped by a combination of their work experiences and personal traits has resulted from the interaction of various aspects, which are represented by machine learning algorithms.

The knowledge acquired in this module was diverse. It became evident that comprehending the data and the issue at hand is just as important to machine learning as choosing the best algorithm. As a crucial component of model optimization, hyperparameter tuning shows that even minor adjustments to parameters can result in better model performance.

The module really advanced my knowledge of machine learning. It emphasized the significance of data pre-processing and the influence of feature selection on model results. Additionally, it illustrated the value of ensemble approaches and the trade-offs between various algorithms. Confusion matrices and ROC curve construction are practical applications that help to ground academic knowledge in decision-making and real-world practice.

To sum up, the study demonstrated how machine learning requires a careful balance between theory and experience. In order to fully utilize machine learning algorithms, it stressed how iterative the modelling process is and how constant learning and adaptation are essential. This practical experience has strengthened my understanding of fundamental machine learning ideas and equipped me to take on increasingly challenging data-driven problems in the future.

References:

Hotz, N. (2018) *What is CRISP DM?*, *Data Science Process Alliance*. Available at: https://www.datascience-pm.com/crisp-dm-2/ (Accessed: 07 January 2024).