**RESEARCH**

**COMP20121**: Foundations of Artificial Intelligence and Machine Learning

**Title:**  
Analysis of the 2024 Stack Overflow Developer

**Student Name:** **Irene Christabel Ogbomo**

**Student ID:** T0404237

**Module Leader:** Jun He

**TABLE OF CONTENT**

1. **Introduction**

* Coursework Objectives…………………………………………………………………………3
* Data mining, Machine learning and Crisp-Dm………………………………………...4
* Comparison with Alternative Data Mining Methodologies …………………….4
* Dataset and Programming Language……………………………………………… 5

1. **Data Understanding, Data Preprocessing and Exploratory Data Analysis**

* Feature Selection and Rationale………………………………………………….6
* Data Quality………………………………………………………………………………….7
* Missing Values…………………………………………………………………………….8
* Identifying and Correcting Erroneous Values……………………………….9
* Outliers Detection and Capping………………………………………………….9
* Data cleaning……………………………………………………………………... 9
* Descriptive Statistics…………………………………………………………………. 10
* Exploratory Data Analysis (Features Selection and Comparison) ………………... 10
* **C**omparison of Relationship between Developer Type and Professional Experience……...17

1. **Cluster Analysis Process and Evaluation………………………………………18**
2. **Machine Learning……………………………………………………………………….22**

* Flowchart…………………………………………………………………………………………………22
* **The Workflow for Machine Learning………………………………………………………22**

1. **Evaluation of Machine Modelling…………………………………………………….26**
2. **Conclusion and Reflection…………………………………………………………………29**
3. **References ………………………………………………………………………………………… 31**

**INTRODUCTION**

The global demand for software engineers is increasing, thereby impacting labour market trends, job satisfaction, and income distribution. Understanding these dynamics is crucial for individuals seeking career advancement and for organizations aiming to attract and retain top talent. To better understand these trends, this coursework analyses the 2024 Stack Overflow Annual Developer Survey, a comprehensive dataset comprising 65,437 responses from over185 countries (Stack Overflow, 2024). The objective of this project is to employ data mining and machine learning methodologies to identify significant trends among developers, categorize them into pertinent groups, and develop predictive models to determine the factors influencing high-earning software professionals. The findings are expected to provide valuable applications for both individuals and organizations, as well as data-driven insights into the developer ecosystem. The primary objective is to predict which developers earn more than the median salary, defined as "high-income" developers, by utilizing various machine learning models and techniques such as exploratory data analysis (EDA), clustering, and classification. Through the analysis of the Stack Overflow Developer Survey dataset, I hope to obtain a thorough grasp of the machine learning procedures utilised in data mining. Exploratory data analysis (EDA), data cleansing, the CRISP-DM technique, and applying machine learning models to forecast high-paying developers are all things I will learn. Along with assessing model performance, I will also find data-driven insights into the variables affecting software developer compensation. In the end, this research will improve my abilities in data analysis and machine learning and provide useful applications for recruiters and career progression in the IT sector.

**Coursework Objectives**

This project uses a systematic machine learning approach to the Stack Overflow dataset with three main activities in focus:  
1. Exploratory data analysis (EDA):

* Recognising trends in the backgrounds, career routes, and incomes of developers.
* Understanding dataset properties and choosing pertinent features.

2. Clustering Analysis:

* Applying clustering algorithms (such as hierarchical clustering and k-Means) to divide developers according to common characteristics.
* Examining cluster features and their effects.

3. Classification Modelling:

* Creating machine learning models to forecast if a developer makes more than the median salary.
* Comparing several categorisation techniques, such as k-Nearest Neighbours, Decision Trees, and Logistic Regression.
* Models are evaluated using ROC curves, F1-score, recall, accuracy, and precision.

**Data Mining, Machine Learning, and CRISP-DM**

Data mining is the process of discovering patterns, trends, and useful information from large datasets using techniques like machine learning, statistical analysis, and database systems. It helps in decision-making by extracting valuable insights from raw data.

In machine learning, algorithms are developed to facilitate data-driven decision-making, whereas in data mining, significant patterns are extracted from vast datasets (Fayyad, Piatetsky-Shapiro & Smyth, 1996). The CRISP-DM (Cross-Industry Standard Process for Data Mining) framework will be utilised to organise this study because it is generally considered a flexible and methodical technique for machine learning projects (Wirth & Hipp, 2000).

CRISP-DM has six essential stages as shown in Figure 1:

1. Business understanding, which includes defining goals and comprehending the dataset.  
2. Analysing and visualising dataset properties is known as data understanding.  
3. Preparing the data: cleaning, converting, and choosing pertinent features.  
4. Modelling: Using clustering and classification methods from machine learning.  
5. Evaluation: Evaluating the model's performance and improving tactics.  
6. Deployment: Gathering information and coming to insightful judgements.

**Figure 1**

Diagram of a diagram of data

Description automatically generated

***Figure 1: CRISP-DM Methodology Flowchart- this show better illustrate how the CRISP-DM methodology will be applied in this coursework***

**Comparison with Alternative Data Mining Methodologies**

There are various frameworks available for organizing data mining projects, such as:

1. SEMMA (Sample, Explore, Modify, Model, Assess): Created by SAS, SEMMA focuses on model development but does not include a dedicated phase for understanding the business context, which makes it less suitable for exploratory research (SAS Institute, 2020).

2. KDD (Knowledge Discovery in Databases): Although it is thorough, KDD does not clearly outline iterative evaluation processes, which limits its flexibility for enhancing models through multiple iterations (Fayyad et al., 1996).

3. Agile Data Science: This approach is adaptable and quick, but it lacks standardization, which can result in inconsistent outcomes in structured machine learning projects (Russom, 2011).

In comparison to these frameworks, CRISP-DM is the most appropriate choice for this coursework due to its structured yet adaptable workflow. It strikes a balance between understanding business needs, maintaining technical rigor, and allowing for iterative improvements, making it well-suited for data-driven machine learning applications.

**Implementation Of CRISP-DM In This Coursework**

The CRISP-DM framework will be utilized in this project as follows:

* Business & Data Understanding:

Examine the Stack Overflow Developer Survey dataset to pinpoint the main factors that affect developer salaries.

* Data Preparation:

Address missing data, encode categorical variables, and standardize numerical features.

* Modelling:

Utilize k-Means and hierarchical clustering to categorize developers into separate groups.

Develop classification models (such as Decision Trees, Logistic Regression, and k-NN) to forecast high-income developers.

* Evaluation & Interpretation:

Evaluate models using performance metrics like accuracy, precision, recall, F1-score, and ROC curves.

Compare the effectiveness of the models and extract significant insights.

**Dataset and Programming Language**

This coursework employs the 2024 Stack Overflow Annual Developer Survey, which includes:

* Developer demographics (age, location, experience).
* Preferences for programming languages.
* Employment trends and salary distributions.

The analysis will be performed using Python, a popular programming language for machine learning and data analysis. Key libraries include:

* Pandas & NumPy for data preprocessing.
* Matplotlib & Seaborn for data visualization.
* Scikit-learn for machine learning modelling and assessment.

By implementing a systematic data mining approach with Python-based machine learning methods, this coursework aims to provide valuable insights into developer career trends, salaries, and workplace dynamics.

**DATA UNDERSTANDING, DATA PREPROCESSONG AND EXPLORATARY DATA ANAYLSIS**

The dataset utilized for this coursework is the 2024 Stack Overflow Annual Developer Survey, featuring 65,437 responses from developers in over 185 countries. It encompasses different facets of developers' experiences, such as their job positions, years of experience, salaries, educational backgrounds, and technology choices. The aim of this analysis is to investigate the dataset, identify significant patterns, and create predictive models to categorize developers into income brackets.

The dataset aims to capture the experiences, challenges, and backgrounds of developers worldwide, making it a valuable resource for understanding trends in the software development industry. Key features include:

* Demographic information such as age, gender, and country of residence.
* Professional experience including job roles, years of experience, and educational background.
* Technological preferences like preferred programming languages and tools used.
* Compensation details including annual salary, compensation by location, and industry.

The dataset is structured into various columns, each representing a unique feature of a developer's profile. These features are critical for understanding the factors influencing developer income, which is the primary focus of this analysis.

**Feature Selection and Rationale**

In the development of robust machine learning models, the identification of pertinent features is crucial. The features outlined below have been chosen due to their anticipated influence on the income of developers:

|  |  |  |  |
| --- | --- | --- | --- |
| FEATURES SELECTED | DESCRIPTION | DATA TYPE | JUSTIFICATION |
| YearsCodePro | Number of years spent working as a software developer professionally. | Categorical | Senior developers often make more money, and experience is a major factor in determining pay. |
| Employment | Employment kind (contract partime, fulltime etc) | Categorical | This feature is important since full-time workers usually make more money than contract or part-time developers. |
| MainBranch | Focus of a developer (e.g fullstack, frontend, backend) | Categorical | Because certain roles (like full stack) may pay more, specialisation in a certain field frequently corresponds with salary. |
| OrgSize | The developer's employer's size (small, medium, or large). | Categorical | This aspect is significant because larger companies might provide greater pay and extra benefits. |
| EdLevel | highest degree of education attained by the developer (e.g., bachelor's, master's, PhD). | Categorical | In the tech sector, higher-paying positions are frequently associated with higher-educational attainment. |
| ConvertedCompYearly | The developer's yearly salary, adjusted for exchange rate fluctuations. | Numerical | This is the analysis's target variable, which is utilised to forecast developers with high incomes. |
| Country | Nation of residency of the developer. | Categorical | Salary is highly influenced by geographic location, with developers in wealthier nations making more money. |
| DevType | Role or type of work for the developer (e.g., frontend, backend, or full-stack developer). | Categorical | Salary is frequently influenced by job role; full-stack developers typically make more because of their broad skill set. |

**Data Quality**

It is crucial to make sure the data is clean and ready for additional analysis while examining the dataset. These covers dealing with outliers, addressing missing numbers, and spotting incorrect entries. The procedures used to clean and arrange the data are summarised below.

**Missing Values**

During the dataset cleaning procedure, missing values were found in several important columns. First, the dataset was cleaned. Here's how these values were handled:

|  |  |  |
| --- | --- | --- |
| Column | Missing values | Handling Method |
| YearsCodePro | 90 | Imputed with the mode of column, The choice of mode imputation is particularly useful for categorical variables, as it helps maintain the existing distribution without introducing new biases. |
| OrgSize | 25 | Imputed with the mode of column, representing the most common organizational size in the dataset. |
| DevType | 32 | Imputed with the mode of column. The most frequent developer roles in the dataset were used to fill the missing entries. This helps ensure that the imputed values align with the general trends in the data. |
| ConvertedCompYearly | 0 | No missing values after table was dropped, as salary data is critical for analysis and rows without salary information were excluded. |

**Figure 3**

A screenshot of a computer error

Description automatically generated **figure 2** A screenshot of a computer code

Description automatically generated

|  |
| --- |
| *Figure 2: Before using handling method (mode)* |

|  |
| --- |
| *Figure 3: After using handling method (mode)* |

By removing rows containing missing salary data and imputing missing values for category columns, the dataset was returned to its whole form, enabling insightful analysis free from the biases associated with missing data.

**Identifying and Correcting Erroneous Values**

Handling Inaccurate Salary Entries: Several data points, including values as low as 1 USD, were found to be incorrect in the "ConvertedCompYearly" (annual salary) column. To preserve the integrity of the dataset, these extreme values were handled as part of the outlier detection procedure since they were regarded as outliers.

**Outliers Detection and Capping**

Outliers have the potential to skew the analysis's findings because they are far outside of the predicted range. Unusual high or low pay numbers were noted as outliers in the "ConvertedCompYearly" column. This was resolved by using the Interquartile Range (IQR) approach, in which values that were outside of the lower bound (Q1 - 1.5 \* IQR) and higher bound (Q3 + 1.5 \* IQR) were referred to as outliers.

Extreme values were found and then capped using the following steps:

* Calculate Q1 (the 25th percentile) and Q3 (the 75th percentile) to evaluate the data's distribution.
* Calculate the IQR, or the range for usual values, by dividing Q3 by Q1.
* Define lower and upper bounds; The lower bound was calculated as **Q1 -1.5 X IQR** and the upper bounds **as Q3 + 1.5 X IQR**

The box plot showed many extreme outliers prior to capping, which distorted the salary distribution (see Figure 4). Following capping, the wage values were restricted to a more realistic range, thus controlling outliers (see Figure 5). To ensure data integrity for subsequent analysis and modelling, this procedure minimised the impact of extreme values while preserving the general distribution.

|  |
| --- |
| ***Fig 4: Box plot before capping outliers, showing numerous extreme values*** |

|  |
| --- |
| ***Fig 5: Box plot after capping outliers, showing the salary distribution within a reasonable range.*** |

A graph with numbers and lines

Description automatically generated A graph showing a number of blue squares

Description automatically generated with medium confidence

**Figure 4. Figure 5**

**Data cleaning**

Ensuring the integrity of the dataset is essential for accurate analysis. Following the resolution of missing values, the rectification of erroneous entries, and the management of outliers, the dataset was validated utilizing the df\_selected. info () method, which confirmed the following:

Non-Null Entries: All 23,435 entries across columns are non-null, signifying successful management of missing data and ensuring the dataset is comprehensive.

Data Types: The dataset comprises 8 columns; 7 categorical variables (e. g., YearsCodePro, EdLevel) classified as objects, and ConvertedCompYearly classified as float64 for continuous data. This correct assignment of data types guarantees appropriate statistical analysis. The df\_selected. info () method ascertained the absence of missing data and validated the structure of the dataset, rendering it ready for further analysis. As depicted in Figure 6, this cleaning process preserves the reliability of the dataset and prepares it for modelling and exploration. With these validations in place, the dataset is now prepared for more in-depth analysis.

A screenshot of a computer

Description automatically generated**Figure 6**

|  |
| --- |
| ***Figure 6: Output from df\_selected.info () confirming the absence of missing values and appropriate data types after data cleaning*** |

**Descriptive Statistics**

Descriptive statistics were computed for important numerical columns, especially "ConvertedCompYearly" (annual salary), to evaluate the efficacy of the cleaning procedure. The cleansed dataset yielded the following statistics:

1. There were 23,435 entries.

2. The average salary is $77,586.77.

3. $58,421.34 is the standard deviation.

4. The minimum wage, following outlier capping, is $1.

5. Maximum Salary (after outlier capping): $220,860.75

6. $62,712 is the 25th percentile (Q1).

7. Median (50th Percentile): $65,000

8. $107,971.50 is the 75th percentile (Q3).

These figures demonstrate how the wage data now falls within a more realistic and fair range following the cleaning procedure. Outliers' impact was successfully reduced by capping them, guaranteeing that the dataset is suitable for additional study (see Fig 7)

|  |
| --- |
| ***Figure 7: Descriptive statistics of "ConvertedCompYearly" after cleaning, showing a more realistic salary distribution.*** |

A screenshot of a computer

Description automatically generated ***Figure 7***

**Exploratory Data Analysis**

Exploratory Data Analysis is an essential phase in data analysis that entails examining the data both statistically and visually to find underlying trends, patterns, and connections. It aids in comprehending the dataset, spotting irregularities, and verifying hypotheses, all of which ultimately direct additional research and model creation. EDA is utilized in this study to examine the ways in which different elements, including hybrid working models, affect the target variable, average income, inside businesses.

**Country**

Country is one of the features examined by EDA, and it is crucial to comprehending how income is distributed among various geographical areas. Two main visualisations that illustrate the study of this feature are a bar plot that compares the average yearly income by nation and a pie chart that displays the dataset distribution by country. The pie chart (Figure 1) provides an overview of how the dataset is distributed across various countries. It clearly shows the distribution of countries in the dataset, with "Others" accounting for 61%. India has only 4%, compared to 20%, 9%, and 6% for the USA, Germany, and the UK, respectively. There is a skewed representation in the chart, with a significant amount of data falling under the "Others."

A pie chart with different colored circles

Description automatically generated**figure 8**

|  |
| --- |
| ***Figure 8: Pie chart showing the distribution of data points by country.*** |

**Average Annual Salary by Country**

The bar plot (Figure 2) contrasts the average yearly wage in other nations to further examine the connection between average income and country. The following insights are revealed by the analysis: With a range of $53,703 to $128,874, the average yearly pay in Germany is $77,054. The lowest average wage is $30,513 in India, with a wide range of $7,178 to $336,716. Although the range is incredibly large, ranging from $24,000 to $2, the average wage in the others group, which includes nations not specifically mentioned, is greater at $65,366.

A graph of blue and white bars

Description automatically generated**figure 9**

**Education Level (Edlevel)**

The analysis of the EdLevel is represented by two key visualizations: a count plot showing the distribution of education levels and a bar plot comparing average annual salary by education level. This count plot shows how the dataset is distributed across education levels. The most prevalent education levels in the dataset are bachelor’s and master’s degrees, while the smallest number of individuals have a Primary/elementary school education.

A screenshot of a computer

Description automatically generated***figure 10***

**Average Annual Salary by Education Level**

A bar plot of the average yearly wage for different educational levels is shown in Figure 11. Important conclusions include:  
1. The greatest average wage is earned by those with master's degrees.  
2. Professionals with professional and doctoral degrees are also paid competitively.  
3. Earnings for those with bachelor’s and associate degrees are comparatively high.  
4. Secondary school graduates and people with some college or university education but no degree make less money.

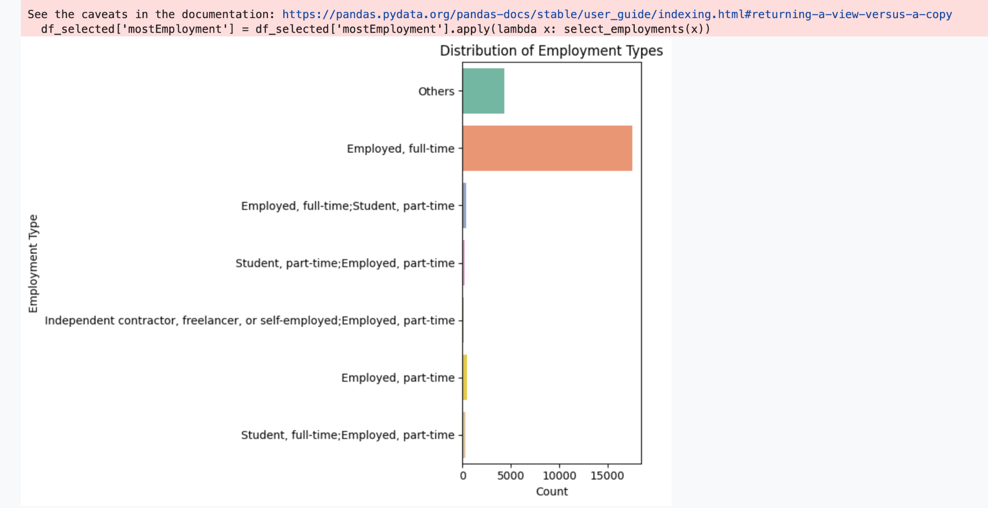
A graph with blue bars

Description automatically generated with medium confidence ***figure 11***

This analysis reveals a strong correlation between educational attainment and income. Individuals with higher educational qualifications, particularly Master's and Doctoral degrees, tend to earn higher salaries and supporting the notion that education is a key factor influencing earnings.

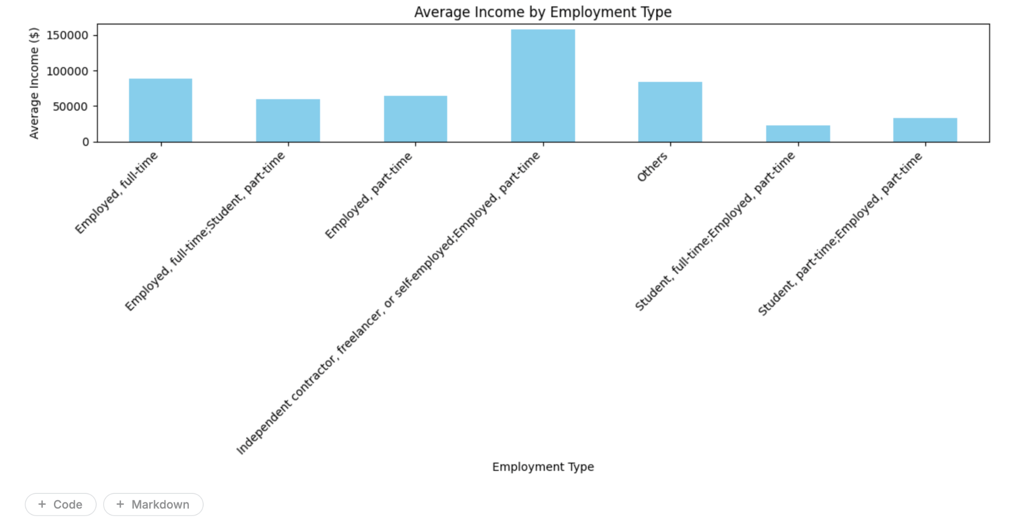
**Employment**

The Employment features provide insights into the relationship between average income and various job statuses, as shown by two visualizations: a count plot for employment type distribution and a bar plot comparing average income by occupation type. The distribution of employment categories is illustrated in Figure 12, the count plot, where full-time employment is the most prevalent. Other categories include Student, part-time; Employed, part-time; Student, part-time; Employed, part-time; and employed, full-time. Additionally, there are self-employed, freelancers, and independent contractors, as well as part-time employees with flexible schedules. The "Others" category contains unspecified job types. The count plot reveals a clear trend toward full-time employment, with Employed, full-time being the most common category, followed by smaller groups in other employment types.

 ***figure 12***

**Average Annual Income by Employment Type**

The study's bar plot indicates that full-time, employed people make the highest average salary, which is $100,000(figure 13). Full-time employees and part-time students make about $70,000, which is considered a moderate-income level. Employees who work part-time for employers make an average of $45,000, while students who work part-time for employers make the least, at about $30,000. The average salary for independent contractors, freelancers, or part-time self-employed people is $50,000. With an average income of about $25,000, the others category has the lowest income, indicating a substantial income disparity among other job categories.

***figure 13***

**Organisation Size (OrgSize)**

The OrgSize (Organisation Size) feature is a crucial component in comprehending how a company's size can affect the average salary of its workers. The relationship between organisation size and income levels can be better understood by examining the distribution of organisation sizes and contrasting average wages across different groups. According to the study, the count plot (Figure 14) displays the distribution of organisation sizes. There are a lot of medium-sized businesses, as seen by the largest categories being those with 20–99 employees and 100–499 employees. There are less people working as freelancers or sole proprietors, according to the "Just me – I am a freelancer, sole proprietor" category. Larger organisations are less prevalent; examples include those with 1,000–4,999 employees, 10,000+ employees, and 500–999 employees. Smaller groups, such as "I don't know" and "2 to 9 employees," indicate even fewer people, with the latter expressing a lack of knowledge over the size of the company.

A screenshot of a graph

Description automatically generated **figure 14**

**Average Income by Organization Size**

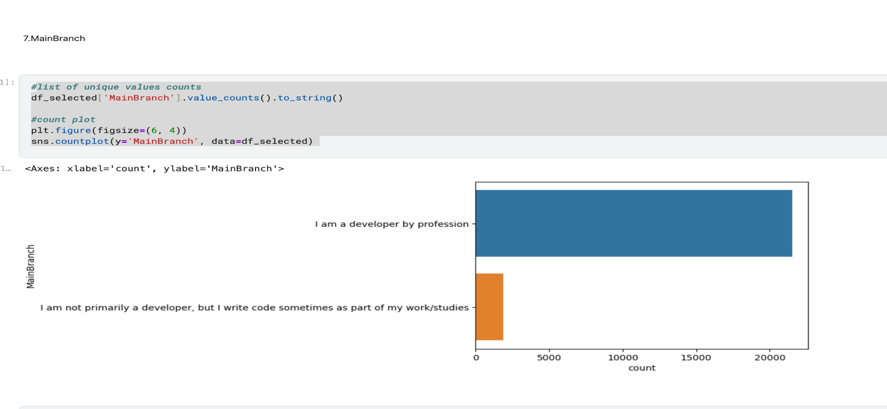
According to the study, larger companies often pay greater salaries (see Figure 15 a bar plot comparing average yearly income by organisation size). The highest average salary is $120,000 for workers in companies with 10,000 or more employees, followed by $95,000 for workers in companies with 1,000–4,999 employees. Employees at businesses with 100–499 workers make $80,000, while those in businesses with 5,000–9,999 workers make $85,000. Freelancers and sole entrepreneurs make the lowest salaries, at $50,000, while smaller businesses, including those with 20 to 99 staff, make about $70,000**.**

**A graph with blue bars

Description automatically generatedfigure 14**

**Main Branch**

To determine whether a person is a professional developer or occasionally codes as part of their career or education, the research looks at the MainBranch feature. The count plot in Figure 15 displays the responses' distribution. People who identified as "I am a developer by profession" make up the largest group; they represent professionals whose main responsibility is development. "I am not primarily a developer, but I write code sometimes as part of my work/studies," the second category, is representative of people who occasionally write code in roles that are not development related. According to the count plot, the majority of the dataset's members are professional developers, with a smaller percentage occasionally using coding for work or education.

 **figure 15**

**Average Income by Main Branch**

The second visualization, Figure 16, a bar plot, compares the average annual income for the two groups identified in the MainBranch feature. The group "I am a developer by profession" has a higher average income of around $85,000, reflecting the earnings of professional developers. In contrast, the group "I am not primarily a developer, but I write code sometimes as part of my work/studies" earns significantly less, with an average income of $55,000. The bar plot clearly shows that professional developers earn considerably more than those who code occasionally in other roles, highlighting the income disparity between these two groups.

A blue and white bar graph

Description automatically generated**figure 16**

**Year Of Professional Experience (YearCodePro)**

The Years of Professional Experience (YearCodePro) feature helps us understand how professional experience correlates with income. By examining how years of experience relate to salary levels, we can gain insights into how career progression and experience influence earning potential. The analysis of the YearCodePro feature clearly illustrates the positive relationship between years of professional experience and income. As shown in both the pie chart(fig18) and bar plot(fig19), most individuals in the dataset have between 1 and 10 years of experience, and their average income is around $55,000. As experience increases, so does the average income, with individuals who have 20+ years of professional experience earning an average of $120,000.

The scatter plot adds an additional layer of insight, showing a weak upward trend between years of experience and income, but also revealing significant outliers and variability in the data(fig20). This suggests that while professional experience is an important factor in determining income, there are other variables at play that contribute to high earnings in certain cases.

A colorful pie chart with numbers

Description automatically generated A graph of a bar chart

Description automatically generated with medium confidence**figure 19**

|  |
| --- |
| *Figure 18: showing the distribution of years CodePro in dataset* |

|  |
| --- |
| *Figure 19: Bar plot comparing average income by years of professional experience.* |

**Figure 18**

|  |
| --- |
| *Figure 20: Scatter plot of income vs. years of professional experience.* |

A graph of a person with a number of blue dots

Description automatically generated with medium confidence**figure 20**

**Developer Type**

The analysis of the DevType feature highlights significant trends in the distribution of developer roles. As shown in fig 21, a bar plot, shows that a few roles dominate the dataset. The full-stack developer role is the most prevalent, making up the largest portion of the data. The desktop or enterprise application developer role also represents a notable share. Meanwhile, specialized roles like data scientist, game developer, and security professional are present but account for much smaller portions. The bar plot emphasizes that full-stack developers are the most common, with fewer individuals in more specialized roles.

**Figure 22** A graph of a number of software types

Description automatically generated

**Developer Type and Salary Range Crosstab**

The research, using a crosstab (Figure 23), examines the distribution of salary ranges across different developer roles. The analysis shows that full-stack developers represent the largest group in the 50K-100K and 100K-150K salary ranges. Front-end and back-end developers also show strong representation in the 50K-100K range, with some advancing to the 100K-150K range. Data scientists and security professionals, as more specialized roles, tend to earn higher salaries, with many in the 100K-150K and 150K-200K ranges. The crosstab highlights that full-stack developers primarily occupy the 50K-100K range, while specialized developers like data scientists typically earn higher wages.

A table of numbers and text

Description automatically generated with medium confidence**figure 23**

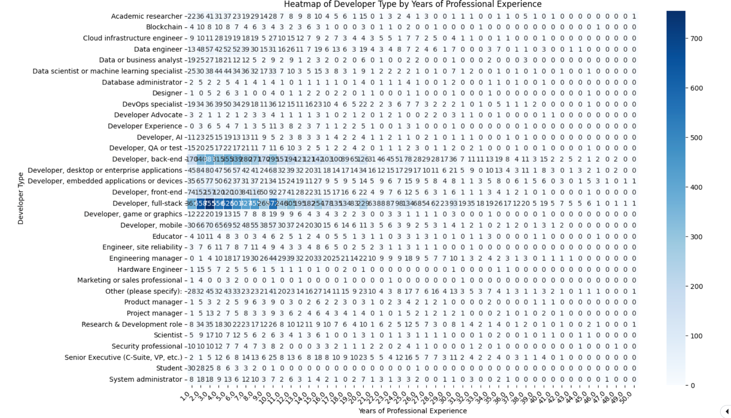
**Comparison of Relationship between Developer Type and Professional Experience**

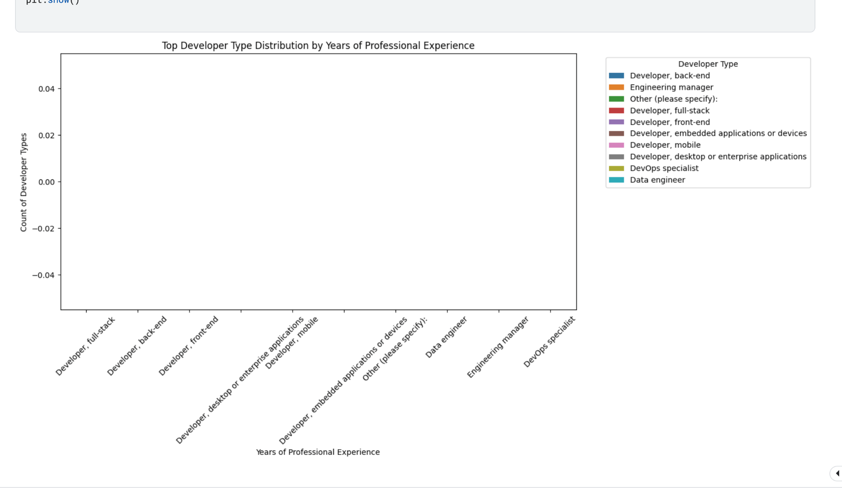
In this analysis, I explored the connection between Developer Type (DevType) and Years of Professional Experience (YearCodePro) to understand how developer roles evolve over time. By examining how different developer roles are distributed across various levels of experience, we can identify patterns in career progression and specialization.

The heatmap (Figure 24) reveals that full-stack developers are most prevalent in the 0-5 years and 6-10 years’ experience ranges. This suggests that newcomers in the field often start with broad roles, such as full-stack development, where they handle both front-end and back-end tasks. As their careers progress, we notice a shift towards more specialized roles, such as Data scientists and DevOps specialists, particularly after 10+ years of experience. This indicates that as developers gain more experience, they tend to focus on specific technical areas rather than remaining in generalist roles.

The count plot (Figure 25) further supports these observations, showing that full-stack developers dominate in the early stages of a developer’s career, while roles like engineering managers, product managers, and data engineers emerge as experience grows. These advanced positions require both deep technical knowledge and leadership skills, which are typically honed over 10-20 years in the field.

In conclusion, the analysis demonstrates a clear relationship between Developer Type and Years of Professional Experience. Developers typically start with generalist roles but, as they accumulate experience, they transition into more specialized or leadership positions (Figures 24 and 25).

 *Figure 24: Heatmap showing the distribution of Developer Types by Years of Professional Experience*

 *Figure 25: Count plot showing the distribution of Developer Types by Years of Professional Experience.*

**CLUSTER ANALYSIS PROCESS AND EVALAUTION**

Cluster analysis is an unsupervised machine learning method used to categorize data points into groups with similar characteristics based on specific features within a dataset. This report aims to uncover patterns and groupings in the data, focusing on attributes such as YearsCodePro (years of professional coding experience), DevType (developer type), and OrgSize (organization size). By applying K-Means and Hierarchical clustering techniques, the analysis seeks to identify clusters of individuals with varying professional backgrounds and work environments. The initial step in the analysis involved preprocessing the dataset. The categorical variables, DevType and OrgSize, were converted using One-Hot Encoding. For instance, DevType, which includes categories like full-stack, back-end, and front-end, was transformed into distinct binary columns. Similarly, OrgSize was encoded to generate separate columns for categories such as “2 to 9 employees” and “10,000 or more employees.” This conversion ensured that the categorical variables could be utilized effectively in clustering, without assuming any inherent order between them. The YearsCodePro feature, indicating years of professional coding experience, was retained as a numerical variable. Any missing values were imputed with the median of the column. Once the missing data was addressed, the dataset was standardized using StandardScaler, which adjusted all features to have a mean of zero and a standard deviation of one. This step was essential to prevent the clustering algorithm from being influenced by features with larger value ranges, such as YearsCodePro.

For the clustering process, I selected K-Means as the primary algorithm. The K-Means algorithm iteratively assigns each data point to the nearest cluster centre and updates the cluster centres based on the points assigned to them. This cycle continues until the cluster assignments stabilize.

To identify the optimal number of clusters, I applied the Elbow Method separately to the High- and Low-income groups. This method involves plotting the distortion (the sum of squared distances between data points and their assigned cluster centres) for different values of k. As shown in Figures 18a and 18b, I generated two separate elbow plots for the High- and Low-income clusters.

For Cluster 0 (High income), the Elbow Plot (Figure 26a) showed that the optimal number of clusters was k = 2, as the distortion curve started to flatten at this point, suggesting that adding more clusters would not significantly reduce the distortion.

For Cluster 1 (Low income), the Elbow Plot (Figure 26b) similarly indicated that k = 2 was the optimal choice, with the distortion curve levelling off after this value.A graph with a line and numbers

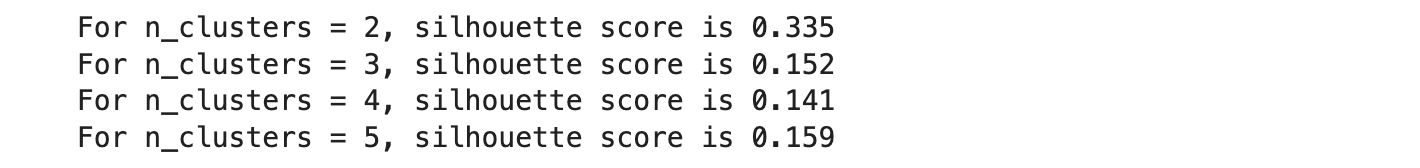
Description automatically generated A graph with numbers and a line

Description automatically generated

*Figure 26a: Elbow method showing the optimum number of clusters for High-income group.*

*Figure 26b: Elbow method showing the optimum number of clusters for Low-income group.*

Next, I calculated the Silhouette Score for each value of k to further validate the choice of k = 2. The silhouette score assesses how similar a data point is to its own cluster compared to other clusters. A higher silhouette score indicates more distinct and well-defined clusters. As shown in Figure 27, the silhouette score was highest for k = 2, confirming that the two-cluster solution was the most optimal.

*Fig 27: Silhouette score analysis for optimal k = 2.*

After determining the optimal number of clusters, I applied K-Means clustering to divide the data into two clusters. To facilitate visualization, I used Principal Component Analysis (PCA) to reduce the data to two dimensions. The PCA plots in Figures 28 and 29 clearly demonstrate a distinct separation between the two clusters, validating the effectiveness of the K-Means algorithm.

A chart of a number of dots

Description automatically generated with medium confidence A chart with a number of dots

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated A chart with a number of dots

Description automatically generated with medium confidence

***Figure 28: PCA of cluster-high Figure 29: PCA of cluster-low***

To further confirm the results from K-Means, I also applied Agglomerative Hierarchical Clustering using the Ward linkage method, which aims to minimize variance within clusters. Also as shown in Figures 30, this output suggests that for some data points (e.g., rows 72, 374, and 392), both clustering algorithms assigned them to the same cluster, while for others (e.g., rows 395 and 398), there are differences between the K-Means and hierarchical clustering results. This is typical, as the two clustering methods can produce slightly different results due to their inherent differences in how they define clusters.

***Figure 30: comparison of clustering results for high income and low income based on the elbow method.***

*A white background with black border

Description automatically generated*

*.*

*A white background with black dots

Description automatically generated*

After identifying the clusters, I analysed their characteristics. The YearsCodePro distribution (Figure 31a and 31b) showed that Cluster 0 (High) consists of individuals with significantly more years of professional coding experience, as the median value of YearsCodePro was much higher than that of Cluster 1 (Low). This suggests that Cluster 0 includes senior or specialized developers, while Cluster 1 consists of junior or entry-level professionals.

A graph of a bar chart

Description automatically generated with medium confidence A screenshot of a computer screen

Description automatically generated

***Figure 30: Box plot of YearsCodePro by cluster (K-Means).***

The OrgSize distribution (Figure 32) highlighted further differences. Cluster 1 (Low) was predominantly composed of individuals from smaller organizations, such as those with "2 to 9 employees" and "10 to 19 employees," along with a considerable number of freelancers. This indicates that Cluster 1 represents professionals in more flexible, small-scale environments. Conversely, Cluster 0 (High) included a higher proportion of individuals from larger organizations, such as "100 to 499 employees" and "5,000 to 9,999 employees," suggesting that this cluster represents professionals in larger, more structured companies.

A screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generated

***Figure 30: Organization size distribution by cluster (K-Means).***

The distribution of DevType revealed that Cluster 0 (High) consists mainly of specialized developers, such as back-end and front-end developers, while Cluster 1 (Low) has a larger proportion of full-stack developers. This indicates that Cluster 1 contains more versatile professionals (Figure 33).

A graph with blue and orange bars

Description automatically generated

***Figure 33:* *Top Developer Types by Cluster-high.***

A graph with blue and orange bars

Description automatically generated

***Figure 33:* *Top Developer Types by Cluster-Low.***

Furthermore, the cluster analysis identified two distinct groups: Cluster 0 (High), with experienced professionals from larger organizations and specialized roles, and Cluster 1 (Low), with less experienced, versatile professionals, including freelancers and full-stack developers. Both methods confirmed two clusters as optimal.

**MACHINE LEARNING**

**A diagram of a process

Description automatically generated  
*Figure 34a: Flowchart illustrating machine learning process***

The flowchart illustrates the crucial stages that guarantee the effective deployment and assessment of a model, and it depicts the sequential procedure involved in a machine learning classification problem. Data collection, which involves gathering the required data, comes first in the process. Data preparation comes next. To get the data ready for the model, this step involves addressing missing values, encoding categorical variables, and scaling features. The next stage after preprocessing the data is model selection, in which the best classification methods are selected according to the problem and type of data. Following model selection, the model's performance is evaluated using a variety of metrics, including recall, accuracy, and precision. After the evaluation, the model's parameters are optimized by hyperparameter tuning, which enhances the model's generalization and predictive ability. To assess the model's performance on unseen data, data testing is done on a different dataset after tweaking is finished. The robustness, efficacy, and predictive accuracy of the classification models are guaranteed by this complete approach. These steps are graphically depicted in the flowchart, which also shows how each stage builds upon the others to create an iterative process that aims to get the best possible model performance.

**The Workflow for Machine Learning**

The workflow for machine learning classification begins with understanding the problem and preparing the data. First, I collected the data and performed feature extraction, identifying the key inputs influencing the predictions. Preprocessing followed, with normalization to scale all features uniformly. I used the MinMaxScaler from sklearn to scale the features between 0 and 1, ensuring models like k-Nearest Neighbors (k-NN), which depend on distance metrics, performed optimally.

After preprocessing, I implemented three classification models: k-NN, Logistic Regression, and Decision Trees. The k-NN model classifies data points based on the majority class of their nearest neighbors, achieving an accuracy of 73.28%. Next, I implemented Logistic Regression using the LogisticRegression model from the sklearn library. Logistic regression is a probabilistic model that estimates the likelihood of an outcome based on a logistic function applied to a linear combination of input features. This model is especially useful for binary classification problems, such as predicting whether an individual earns high or low income. I trained the logistic regression model on the same normalized dataset, and after evaluation, the model yielded an accuracy of 82.93%, outperforming k-NN. Logistic regression has the advantage of being interpretable, as the coefficients represent the influence of each feature on the predicted outcome. For instance, the coefficient for *YearsCodePro* was -0.098341, indicating that as the years of experience increase, the likelihood of earning a higher income decreases, all else being equal. This negative coefficient aligns with the observation that senior professionals may have plateaued in their careers, potentially earning less due to career shifts or pursuing different job roles.

I used the sklearn DecisionTreeClassifier for Decision Trees, adjusting the max\_depth parameter to 3 to prevent overfitting and enhance generalization. Decision trees make choices at each node of the tree and divide the data according to feature values. Using the same dataset, I trained the decision tree, and the model's accuracy was 73.22% (figure 34b). The decision tree model provides interpretability that other models do not, even if the performance was comparable to that of k-NN. Each node in the decision tree represents a choice based on characteristics, making it evident how decisions are formed. For instance, the tree showed that the most crucial characteristics for differentiating between high- and low-income earners were YearsCodePro and OrgSize (organization size). This is useful for comprehending the elements that affect a person's income.

Accuracy, precision, recall, and F1 score were used to evaluate the models. Precision and recall are important when working with imbalanced datasets or when the outcomes of false positives and false negatives differ, even if accuracy is sometimes the clearest evaluation statistic. In this instance, the models were compared based on accuracy. The linear model better reflected the relationship between the characteristics and income categories, as demonstrated by the best results from logistic regression, k-NN, and decision trees.

The performance of the classification models was evaluated using standard metrics such as accuracy, precision, recall, and F1 score. These metrics help to measure the effectiveness of the models, especially in the case of imbalanced datasets or when the cost of false positives and false negatives varies.

The accuracy of a model is the proportion of correctly predicted instances (both true positives and true negatives) among the total instances. It is calculated as:

**Accuracy = TP +TN**

**TP+TN+FP +FN**

where:

* TP = True Positive (correctly predicted positive class),
* TN = True Negative (correctly predicted negative class),
* FP = False Positive (incorrectly predicted as positive),
* FN = False Negative (incorrectly predicted as negative).

Precision measures the proportion of positive predictions that were correct, and it is defined as:

**Precision = TP**

**TP+ FP**

Recall, also known as sensitivity or true positive rate, measures how well the model identifies positive instances, and it is calculated as:

**Recall = TP**

**TP+ FN**

Lastly, the F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics, and it is given by:

**F1 = 2. Precision.Recall**

**Precision+Recall**

These metrics were used to assess the quality of the k-NN, logistic regression, and decision tree models. The precision, recall, and F1 scores give a more detailed understanding of the models' performance, particularly in terms of handling misclassifications, which is crucial when predicting income levels

The Random Forest Classifier achieved an accuracy of 81.13%, placing it as a competitive option for the classification task when compared to other models like k-Nearest Neighbours (k-NN), Logistic Regression, and Decision Trees. Among the models tested, Logistic Regression led with an accuracy of 82.93%, narrowly surpassing Random Forest. This result suggests that for datasets with a linear relationship between the features and the target, Logistic Regression might perform slightly better due to its simpler, probabilistic approach. However, Logistic Regression assumes a linear connection, which may not always hold true for more complex, non-linear datasets.

On the other hand, Random Forest outperformed both k-NN (73.28%) and Decision Trees (73.22%), demonstrating its strength in handling complex, non-linear relationships. One of the key advantages of Random Forest over individual Decision Trees is its ability to mitigate overfitting. While single Decision Trees tend to overfit the training data especially when they grow too deep, Random Forest reduces this issue by using an ensemble of decision trees, each trained on different random subsets of the data. This approach helps the model generalize better, making it more reliable when applied to new, unseen data.

While logistic Regression achieved the highest accuracy, Random Forest stands out due to its ability to manage high-dimensional and noisy data. Its ensemble technique also reduces variance, making it highly effective for large, complex datasets. By maintaining stable performance across various test sets, Random Forest proves to be a valuable tool for classification tasks, where both robustness and accuracy are crucial. Despite a small difference in accuracy compared to Logistic Regression, Random Forest remains an invaluable model for real-world applications.A diagram of a structure

Description automatically generatedA screenshot of a computer code

Description automatically generated

***Figure 34b: Decision Tree structure and classification accuracy.***

**EVALUATION OF MACHINE MODELLING**[**¶**](https://kkb-production.jupyter-proxy.kaggle.net/static/assets/jupyterlab-v4/jupyterlab-index-01cbbd656c0eb8a0605e.html?session=eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.#EVALUATION-OF-MACHINE-MODELLING)

To assess and compare the performance of various machine learning models, multiple evaluation metrics were employed, including accuracy, precision, recall, F1-score, confusion matrix, and the ROC curve. These metrics offer valuable insights into the models' effectiveness, especially in distinguishing between two income categories: high and low. The analysis involved three models; Logistic Regression, Decision Trees, and k-Nearest Neighbours (k-NN) which were trained and tested on a dataset aimed at predicting whether an individual earns above or below the median income threshold. The model’s performance was evaluated based on their ability to accurately classify individuals into these income groups.

One of the most useful performance metrics in this analysis was the ROC curve (Receiver Operating Characteristic curve), which illustrates the balance between the true positive rate (TPR) and the false positive rate (FPR) as the decision threshold varies. A model with a higher ROC curve is better at distinguishing between the two income categories, and its effectiveness can be quantified using the area under the curve (AUC). The AUC score ranges from 0 to 1, with higher values indicating stronger model performance.

Among the models tested, Logistic Regression performed the best, achieving an AUC of 0.90 (as shown in Figure 35). This suggests that it was the most effective at correctly classifying both high and low-income individuals. Its strong performance is reflected in its high true positive rate and low false positive rate. The Decision Tree model followed with an AUC of 0.80, showing slightly weaker classification ability than Logistic Regression. This drop in performance may be due to overfitting, as Decision Trees tend to create complex structures that fit the training data well but struggle with new, unseen data. Lastly, the k-NN classifier had the lowest AUC of 0.79, indicating more difficulty in correctly classifying low-income individuals compared to Logistic Regression.

A screen shot of a graph

Description automatically generated

***Figure 35: ROC CURVE COMPARISON***

Beyond the ROC curve, the classification report provides a detailed breakdown of the model’s performance using precision, recall, F1-score, and support. Precision measures the proportion of correctly predicted positive cases, while recall evaluates how well the model captures all actual positive instances. The F1-score balances these two metrics, and support represents the number of actual occurrences of each class in the dataset.

Logistic Regression delivered strong results in the classification report. As shown in Figure 36, the model had a precision of 0.69 for high-income individuals and a recall of 0.85, resulting in an F1-score of 0.76. This means that while the model was slightly less accurate in identifying high-income individuals, it effectively captured most of them. For low-income individuals, it achieved a precision of 0.80 but had a recall of 0.61, leading to an F1-score of 0.69. The lower recall suggests that some low-income individuals were misclassified as high-income, increasing the number of false negatives. However, the model still maintained a good balance between precision and recall, especially in identifying high-income individuals, which is particularly important in datasets with class imbalances.

A screenshot of a computer

Description automatically generated

***Figure 36: Classification Report***

The Decision Tree classifier produced similar results, with a precision of 0.69 for high-income individuals and 0.80 for low-income individuals. The recall was 0.85 for high-income and 0.61 for low-income, mirroring the performance of Logistic Regression. This is expected since Decision Trees tend to overfit, making them more prone to misclassifying low-income individuals. Overfitting occurs when the tree divides the data into very specific groups, making predictions less reliable for unseen data.

The k-NN classifier followed the same trend, achieving precision scores of 0.69 for high-income and 0.80 for low-income individuals, with recall values of 0.85 and 0.61, respectively. However, k-NN performed slightly worse than both Logistic Regression and Decision Trees, as seen in its lower AUC and classification report metrics. This may be due to k-NN’s sensitivity to the selection of k (the number of neighbours used for classification), which affects whether the model overfits or underfits. Additionally, k-NN tends to struggle in high-dimensional datasets, where many features can reduce its classification accuracy and increase computational complexity.

The confusion matrix is a useful way to assess a model’s classification performance by comparing its predictions with actual values. It provides insights into the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). As shown in Figure 37, Logistic Regression had 2848 true positives for high-income individuals and 2816 for low-income individuals, with fewer false positives and false negatives. This suggests that the model correctly classified most individuals in both categories, making it more reliable compared to the Decision Tree and k-NN classifiers. The lower number of misclassifications further supports the conclusion that Logistic Regression had the best overall accuracy among the three models.

The Decision Tree and k-NN classifiers displayed similar confusion matrix patterns, with more errors in predicting low-income individuals. This can be seen in the higher number of false negatives in both models. In the Decision Tree confusion matrix, there were more false positives and false negatives for low-income individuals compared to Logistic Regression, showing the model’s tendency to misclassify this group. The k-NN classifier followed the same trend, further demonstrating its difficulty in accurately identifying low-income individuals. This issue likely stems from k-NN’s reliance on proximity-based classification, making it sensitive to irrelevant or noisy features, which can negatively affect its performance.

A yellow and purple squares with numbers

Description automatically generated

***Figure 37: Confusion Matrix (high and Low income)***

Hyperparameter tuning is a crucial step in optimizing machine learning models to enhance their performance. In k-NN, selecting the right number of neighbours (k) is particularly important. A small k can lead to overfitting, while a large k may cause underfitting. Decision Trees also require careful adjustments, such as controlling tree depth and setting a minimum number of samples for node splits. Limiting tree depth or increasing the required sample size helps reduce overfitting and improves the model’s ability to generalize. Logistic Regression, though less prone to overfitting, still benefits from regularization, with the C parameter controlling its strength.

To find the best settings for each model, hyperparameter tuning was performed using cross-validation. This process helped identify the optimal values, leading to performance improvements, especially for Decision Trees and k-NN, where tree pruning, and k adjustments made a noticeable difference. However, even after tuning, Logistic Regression remained the best model, achieving the highest AUC score and the most balanced precision and recall, making it the most effective choice for this task.

**CONCLUSION**

In summary, this analysis demonstrated the importance of selecting and optimizing the right machine learning model for predicting whether individuals earn above or below a specified income threshold. Of the three models tested: Logistic Regression, Decision Trees, and k-Nearest Neighbours (k-NN). Logistic Regression emerged as the most effective, achieving the highest AUC score of 0.90. This result showed that Logistic Regression was better at distinguishing between high and low-income individuals, with strong recall for high-income individuals and balanced precision across both classes. The confusion matrix further confirmed its strong performance, indicating fewer misclassifications for both income categories.

While Decision Trees and k-NN also provided valuable insights, they had certain limitations. Decision Trees were prone to overfitting, particularly when predicting low-income individuals, and their performance improved only after proper tuning. Similarly, k-NN struggled with low-income classification, with performance heavily influenced by the choice of neighbours. Both models showed improvements after hyperparameter tuning, but Logistic Regression remained the most reliable and accurate choice.

The results underscore the importance of hyperparameter tuning, cross-validation, and evaluating models using multiple performance metrics like the ROC curve, classification report, and confusion matrix. These metrics provided a thorough understanding of each model's strengths and weaknesses. Despite improvements in Decision Trees and k-NN, Logistic Regression consistently outperformed the other models in terms of overall accuracy.

This analysis also highlighted the need for continuous improvement. Like enhancing feature engineering and applying advanced data preprocessing techniques could address challenges such as class imbalance, leading to more accurate and reliable predictions in real-world applications. Ultimately, choosing the right model and fine-tuning it based on dataset characteristics is crucial to achieving the best performance.

Reflecting on this module, I feel I have gained a deeper understanding of machine learning, especially in classification tasks. The skills I’ve developed go beyond technical knowledge and include analytical abilities that allow me to evaluate different models and choose the best one for a problem. I’ve learned how to preprocess data, handle class imbalances, and assess models using metrics like AUC, precision, recall, and F1-score essential skills in data science and machine learning that will be invaluable for my future growth.

A key takeaway from this module was learning how to evaluate and compare models. Observing how small changes in parameters can impact results was insightful and will be crucial for real-world machine learning projects. I’ve also gained experience visualizing performance with tools like the ROC curve and confusion matrix, improving my ability to communicate findings effectively.

Moreover, hyperparameter tuning has made me more confident in optimizing models. Understanding how to adjust parameters like learning rate and tree depth is critical for improving model performance. This module has set a strong foundation for my career in data science and will significantly help in my data analyst internship and third year of degree at university where I’ll be doing more advanced machine learning and get to apply these skills to real-world projects. I also plan to explore advanced techniques like ensemble methods and deep learning to continue expanding my knowledge.

**REFERENCES**

Alan, B. (2024). *Stack Overflow Annual Developer Survey 2024*. [online] Kaggle.com. Available at: <https://www.kaggle.com/datasets/berkayalan/stack-overflow-annual-developer-survey-2024> [Accessed 19 March 2025].

Alvarestech.com. (2025). Available at: <http://alvarestech.com/temp/Industria4.0/2019/(Morgan%20Kaufmann%20Series%20in%20Data%20Management%20Systems)%20Ian%20H.%20Witten,%20Eibe%20Frank,%20Mark%20A.%20Hall,%20Christopher%20J.%20Pal%20-%20Data%20Mining_%20Practical%20Machine%20Learning%20Tools%20and%20Techniques-Morgan%20Kaufmann%20Publisher.pdf> [Accessed 20 March 2025].

CSC380: Principles of Data Science Introduction to Machine Learning. (n.d.). Available at: <http://www.pachecoj.com/courses/csc380_fall21/lectures/mlintro.pdf>. [Accessed 26 March 2025].

Data School (2021). *Plot a confusion matrix*. [online] YouTube. Available at: <https://www.youtube.com/watch?v=QRFMgKdF-Ug> [Accessed 28 March 2025].

Fayyad, U. and Uthurusamy, R. (1996). Data mining and knowledge discovery in databases. *Communications of the ACM*, [online] 39(11), pp.24–26. doi: <https://doi.org/10.1145/240455.240463> [Accessed 20 March 2025].

Nick, T.G. and Campbell, K.M. (2007). Logistic Regression. *Topics in Biostatistics*, 404, pp.273–301. doi: <https://doi.org/10.1007/978-1-59745-530-5_14> [Accessed 27 March 2025].

Hand, D.J. (2007). Principles of Data Mining. *Drug Safety*, 30(7), pp.621–622. doi: <https://doi.org/10.2165/00002018-200730070-00010> [Accessed 14 March 2025].

Raut, R.D., Yadav, V.S., Cheikhrouhou, N., Narwane, V.S. and Narkhede, B.E. (2021). Big data analytics: Implementation challenges in Indian manufacturing supply chains. *Computers in Industry*, [online] 125, p.103368. doi: <https://doi.org/10.1016/j.compind.2020.103368>. [Accessed 14 March 2025].

Shahapure, K.R. and Nicholas, C. (2020). Cluster Quality Analysis Using Silhouette Score. *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. doi: <https://doi.org/10.1109/dsaa49011.2020.00096>. [Accessed 23 March 2025].

Schratz, P., Muenchow, J., Iturritxa, E., Richter, J. and Brenning, A. (2019). Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecological Modelling*, 406, pp.109–120. doi: <https://doi.org/10.1016/j.ecolmodel.2019.06.002>. [Accessed 23 March 2025].

Super Data Science (2023). *The Elbow Method Explained in Less than 5 minutes*. [online] YouTube. Available at: <https://www.youtube.com/watch?v=ht7geyMAFfA> [Accessed 19 March 2025].

‌Vardakas, G., Papakostas, I. and Likas, A. (2024). *Deep Clustering Using the Soft Silhouette Score: Towards Compact and Well-Separated Clusters*. [online] arXiv.org. Available at: <https://arxiv.org/abs/2402.00608> [Accessed 23 March 2025].

www.youtube.com. (n.d.). *StatQuest with Josh Starmer - YouTube*. [online] Available at: <https://www.youtube.com/@statquest>. [Accessed 28 March 2025].