

School of Science and Technology

COMP20121

Foundations of AI & machine learning

From Data to Insights:

Machine Learning and Clustering on the 2024 Stack Overflow Developer Survey

**by**

**Sahadh Fazal Mohamed**

**N1249874**

Table of Contents

[**1. Introduction 3**](#_Toc194489065)

[**2. Data Understanding, Data Pre-processing, Exploratory Data Analysis 5**](#_Toc194489066)

**2.1 Background of the dataset…………………………………………………………………………………………5**

**2.2 Dataset Composition and Feature Selection……………………………………………………………….5**

**2.3 Data Characteristics…………………………………………………………………………………………………..6**

**2.4 Data Cleaning…………………………………………………………………………………………………………….7**

**2.5 Graphical Representation (EDA)………………………………………………………………………………..9**

[**3. Cluster Analysis 11**](#_Toc194489067)

**3.1 What is Clustering…………………………………………………………………………………………………….11**

**3.2 Data Preparation for Clustering………………………………………………………………………………..11**

**3.3 Finding the Ideal Number of Clusters……………………………………………………………………….11**

**3.4 Cluster Classifications and Visualizations………………………………………………………………….12**

**3.5 Summary………………………………………………………………………………………………………………….14**

[**4. Machine Learning for Classification and their Implementation 15**](#_Toc194489068)

**4.1 Logistic Regression…………………………………………………………………………………………………..16**

**4.2 Decision Tree……………………………………………………………………………………………………………17**

**4.3 k-Nearest Neighbors (k-NN)……………………………………………………………………………………..19**

**4.4 Ensemble Method – Voting Classifier……………………………………………………………………….20**

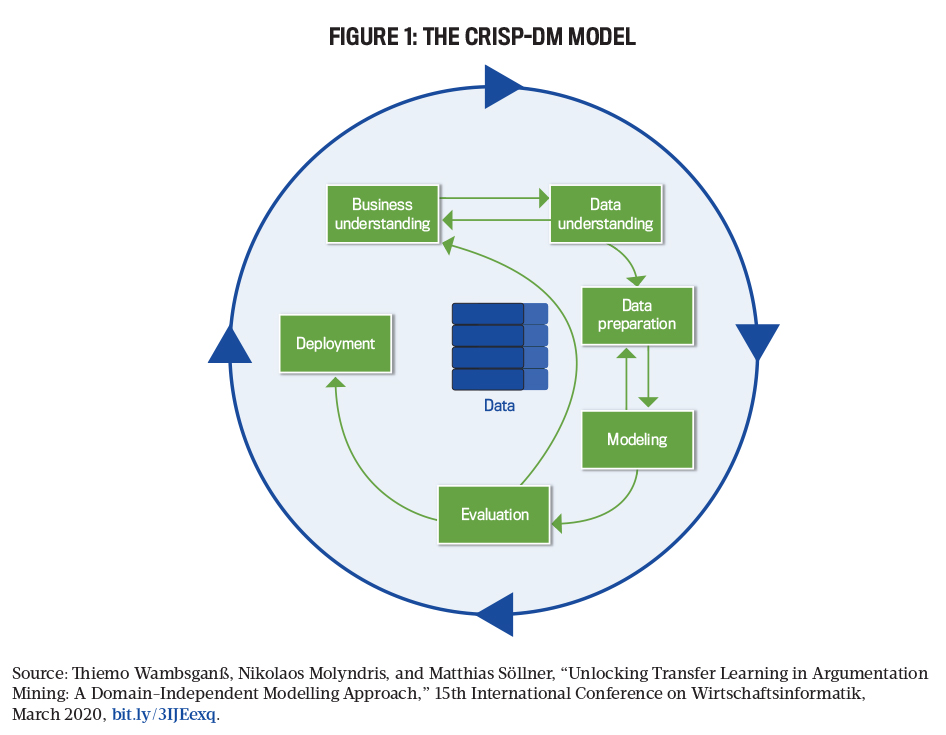
[**5. Evaluation Machine Learning Models 22**](#_Toc194489069)

[**6. Discussions and Conclusions 24**](#_Toc194489070)

[**Chapter 1 References 25**](#_Toc194489071)

[**Chapter 2 Appendix 26**](#_Toc194489072)

# Introduction

Selecting a methodology before planning a project is important for guaranteeing its success. It aids in defining objectives, setting clear goals, and outlining the work scope. According to a 2020 Harvard Business Review data analytics survey of 600 senior-level executives, 55% of organizations believe data analytics is highly significant in decision-making (Lisa S. Haylon, 2023). This is where CRISP-DM, or the Cross-Industry Standard Process for Data Mining, comes in as a rigid framework for planning, executing, and evaluating data mining projects. CRISP-DM is the industry standard today for small and large projects in a broad spectrum of industries due to its adaptability. CRISP-DM was the most common data analytics approach for over two decades. It was constructed by a panel of seasoned data mining experts throughout the late 1990s (Richard O'Hara, 2023). Some other methodologies exist for data analysis development, including SEMMA and KDD. In KDD, a database is employed, and any preprocessing, subsampling, and database transformation are needed to induce knowledge based on measures and threshold specification (Fayyad et al., 1996). On the other hand, SEMMA, which stands for Sample, Explore, Modify, Model, and Assess, is also on the same level as compared to CRISP-DM. The methodology SEMMA was developed by the SAS Institute in 1998 for their data mining software. SEMMA is more SAS-specific and frequently used in conjunction with CRISP-DM. Although SEMMA is less well-known and less frequently used outside of the SAS, it helps develop the model within the SAS environment. While CRISP-DM is extremely comprehensive and documented, all stages are organized and structured, allowing projects to be easily understood and revised (Santos & Azevedo, 2005). Because of the adaptable and thorough framework for machine learning and data mining projects, CRISP-DM will be used for this project to guarantee an organized and structured approach to the data mining process, which will ultimately produce more effective and efficient results. CRISP-DM is a recursive planning, organizational, and execution model of a data mining project. The model has six major phases, as shown in Figure 1. All six phases are crucial to the success of the data mining project and must be appropriately completed to achieve the desired output. To start with, business understanding: the key aim of this phase is to understand the business problem and project objectives. The business understanding phase is critical because it lays the groundwork for the entire project.

***Figure 1: CRISP-DM Model***

It entails creating company goals, defining success factors, and devising a plan to reach the objectives via data analysis. Applying data mining methods to derive meaningful solutions and insights is impossible without in-depth knowledge of the business problem, i.e., finance. The second step is data understanding; it involves collecting the data needed from the second step, which will enhance the efficiency of the data science project. Data can be gathered in the first stage of exploratory data analysis (EDA), and the data is ready and available to be analysed later; conclusions are drawn. Data plots and graphs play a vital role in data visualisation and conveying insights (Sameer, 2024). Followed by the data preparation phase, where data preprocessing and data cleaning are done to render the data quality high and precise enough to analyse. This is a crucial step in ensuring the data is modeling-ready and can provide sound results. The process of data preparation involves all the steps to construct the final dataset from the raw data (Azevedo, Ana & Santos, Manuel, 2008). In data understanding, it's always important to ask questions about the data during each of the tasks. Data preparation includes the activities for defining the data strategy; data management processes, roles, and responsibilities; data life cycle; and architecture. (Fleckenstein & Fellows, 2018). This is followed by the modeling stage, in which statistical and machine learning techniques are applied to processed data to establish a prediction model or infer insight. This phase is crucial to infer sensible information and data-driven decisions based on data analysis. In this phase, various modeling methods are selected and applied. And their parameters are set to optimal values. (Azevedo, Ana & Santos, Manuel, 2008). In the modeling phase, it goes back to data preparation and refining to ensure the accuracy and reliability of the prediction model.

The next final step before the deployment is the evaluation of the model's performance using metrics such as accuracy and precision. This evaluation helps determine the effectiveness of the prediction model and its ability to generalize to new data. This step reviews the business objectives, and it goes back to business understanding to ensure that the model aligns with the goals and requirements of the organization. Finally, in the deployment phase, the project is deployed with a report to assess the success. The objective of the model is to increase the knowledge of the data; the resulting knowledge needs to be organized and presented in a way that is accessible to the customer. (Chapman et al., 2000)

The CRISP-DM methodology is what will be applied in this project to ensure a structured approach to data analysis. The data for this project is the 2024 Stack Overflow Developer Survey data. The dataset contains 65,437 responses from developers across more than 185 countries.

**Business understanding:**

The goal of the project is to analyse the dataset to predict whether a developer is in the high-income category, which is the income > median of ConvertedCompYearly. This analysis provides insights like the factors influencing the developer, like income, which can be useful for businesses and developers.

**Data understanding:**

Our goal is to explore and understand the dataset. The main features include "YearsCode," "EdLevel," "Country," and "ConvertedCompYearly." To understand the different features and to detect outliers, exploratory data analysis (EDA) must be done.

**Data Preparation:**

Cleaning and preprocessing the data is essential before building models. Which involves handling the missing values (NaN), removing outliers, encoding the categorical variables, and normalizing or scaling numerical features.

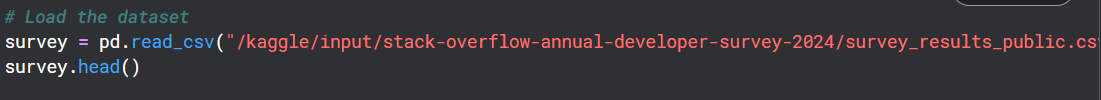
**Insights:** From applying the CRISP-DM to the project, several key insights are derived. These include the identification of the primary drivers of the developer's income (e.g., experience, education), analysis of the salary distribution across various countries, and the comparison of the performance of the predictive models. These can help companies in being able to tailor their recruitment drives and remuneration packages, and the developers can apply these as justifications for making informed career decisions.

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

**2.1 Background of the dataset**

The 2024 Stack Overflow Developer Survey, with more than 65,000 developers from 185 countries answering, was utilized as the source data for this endeavour. Stack Overflow conducts a yearly global survey to observe professional and enthusiast developers' trends and experiences. The data collection provides rich details regarding developers' work, ability, interests, pay, and job satisfaction. An online voluntary survey on Stack Overflow is used to collect the data. As the data collection is self-reported, it may be inconsistent, but understanding developer trends in general is useful.

The dataset contains statistics on a range of topics, including programming languages, tools, and technologies developers use daily. Although providing useful information about the development community, the volunteer collection nature of the data could lead to it not being entirely representative of all developers worldwide. The data also contains information about the employment tasks, experience years, and educational levels of developers. Businesses can use this vast data to make educated decisions based on the trends and preferences of the developer community.



The results obtained from the survey are stored in a CSV file; these results are loaded into the notebook using the .read\_csv method to perform various operations.

**2.2 Dataset Composition and Feature Selection**

There are 83 columns (features) and 65347 rows (responses) in the raw dataset, many of which are sparse, unimportant, or challenging to measure. Seven essential features that have the most ability to affect developer income and work happiness were chosen for our research. We may do more effective data analysis and get significant insights by concentrating on these important characteristics.

The 7 features are as follows:

A screenshot of a computer

AI-generated content may be incorrect.

***Figure 2: Selected 7 features for the coursework***

**2.3 Data Characteristics**

From Figure 2, we can see the 7 features that are selected. These features were chosen based on their potential impact on developer income. From the above figure, we can see the data types, unique values, and missing values for the features. The data type tells us what kind of data is stored in the column, which can be an object (text or mixed values) or int64 or float64 (numerical values). For example, the data type for the "Annual Salary" feature is int64, indicating that it stores numerical values representing the annual earnings of a developer in USD. This is very important to be analysed as the numerical values can be scaled while the categorical values need to be encoded to perform clustering. The unique value is the number of different entries in the column. For example, you can see that the country feature has 185 unique entries, which means that the developers from 185 different countries responded to the survey. The missing values are the number of rows with no data (i.e., NaN) for a specific column. This is necessary as it needs to be handled before analysis.

A screen shot of a computer

AI-generated content may be incorrect.What can we do?

1. We can drop the specific row for that specific column.
2. We can fill them with "unknown."
3. We can impute with a statistic.

***Figure 3: Checking NaN for “Employment”***

That will be clearly explained in the Clustering section.

From the Figure 2 table, you can also check that the feature "YearsCode" datatype is an object, but that creates a question here, as years of coding experience is typically a numerical value.

As from the raw dataset survey before cleaning, "YearsCode" includes values like "Less than 1 year" and "More than 50 years." These are strings, so the Panda Library reads the whole column as an object. After the cleaning, it becomes float64, which is a numerical column with decimal support, because the missing and non-convertible strings become NaN, and it even shows float64 when NaN is present.

Also, if we look at the feature "Employment" in the table, it shows 0 in the missing value column, which means there are no empty (i.e., NaN) values. To ensure whether it's true, we can see that in Figure 3, the output shows 0, which made it easier for us to cluster.

Figure 4 shows a quick summary of the dataset before moving on to the cleaning process. The command used to get that information is "survey.info()". This information shows

1. Index Range—which tells how many rows are in the dataset, which is 65437 entries.
2. A screen shot of a computer code

   AI-generated content may be incorrect.Column Count & Names—114 entries, starting from column ResponseId to JobSat.
3. Data Types—shows whether a column is int64, float64, or object. As shown in the figure, there are 100 object data types, 13 float64 data types, and only 1 int64 data type.

***Figure 4: Dataset Survey Info***

From these 114 features, we are going to take our 7 features to do the necessary operations.

A screenshot of a computer program

AI-generated content may be incorrect.**2.4 Data Cleaning**

Having examined the data and selected features for the coursework, the next step is to clean the data to enable analysis to be conducted. Before we can clean the data, we should identify the number of respondents, missing values, and outliers. Cleaning the data is required as it will rid us of data skew and make our analysis accurate and reliable. Also, removing outliers and missing values will make our results, in general, better. Figure 5 shows the null values for the selected 7 features in which the employment feature has 0 null values. Data is cleaned by dropping the missing value row, handling the categorical field by filling the missing value with "Unknown," and using the arithmetic method like mean or median.

***Figure 5: Null Values before cleaning***

1. Rows with missing target values are dropped

The targeted feature for our project is the "ConvertedCompYearly," which is the annual salary in USD. From Figure 5, we can see that the salary column has 42002 missing values. Since the missing values cannot be used for clustering, we need to clean all the rows with null entries in the column. To clear all the null entries, we can use the command ".dropna()". Figure 6 shows the code snippet that was used in the project.

1. Managing Categorical Fields

A black background with colorful text

AI-generated content may be incorrect.Several categorical columns, including Country, EdLevel, DevType, Employment, and LanguageHaveWorkedWith, had missing values. These were filled with the placeholder "Unknown" to avoid dropping valuable rows. Also, some of these fields contained multi-label responses, like developers choosing more than one role or language. To simplify this data, we only retained the first listed value. This makes clustering and classification easy.

***Figure 6: Dropping Missing Values***

Figure 7 shows the cleaning process for the Employment feature. As we already know that this feature does not have any null values but may contain values that make it difficult for us to cluster, we filled the uncategorized rows with "Unknown." This process goes for the features Country, EdLevel, DevType, and LanguageHaveWorkedWith as well.

1. A screen shot of a computer code

   AI-generated content may be incorrect.Converting Experience into Numeric

If we look in the "YearCode" column, some rows contain values like "Less than 1 year" or "More than 50 years," making it unusable for modeling. Because of this reason, it shows an object in the data type before cleaning. These were converted to numeric values using pd.to\_numeric() with coercion for any non-convertible strings. In Figure 7, missing values were filled using the median years of experience. After cleaning, the data type will be changed to float64 for "YearsCode."

***Figure 7: Cleaning Categorical and Numerical Field***

1. Creating a Binary Target Variable

The goal of this project is to predict whether a developer is a high earner. Since income "ConvertedCompYearly" is a continuous numeric variable, we cannot directly use it for classification unless we convert it. So, we create a binary classification problem by creating a new variable called highIncome. This column assigns the value 1 to developers with earnings higher than the dataset's median annual salary and the value 0 to developers with earnings at or below the dataset's median annual salary.

In Figure 8, we can see the code implementation for this step.

A screen shot of a computer program

AI-generated content may be incorrect.

***Figure 8: Code for “highIncome”***

Figure 9 shows the comparison before and after cleaning of the 7 features. The cleaning process involved handling missing values and outliers. This resulted in a more accurate representation of the dataset for predicting high earners among developers. If we look at the table below, we can see that in the "Missing Values (before)" column, all the 6 features have missing values except one, which is the "Employment" feature. After cleaning the dataset in the "Missing Values (after)" column, all the null entries are now filled with either "Unknown" or the median value. This makes the classification process easier. Also, if you look at the datatype column, in the "YearsCode" column it was an object data type before cleaning, and it changed into float64 after cleaning.

The outliers are detected using the Interquartile Range (IQR) method. Any values 1.5x below the 1st quartile or above the 3rd quartile are flagged. This helps us to understand which numeric columns have extreme values that could affect model training.

A screenshot of a computer

AI-generated content may be incorrect.

***Figure 9: Table showing comparison before and after cleaning***

**2.5 Graphical Representation (EDA)**

This part comprises six visualizations to explore top patterns and associations between the developer survey data. The plots depict income trends by lines of experience, employment type, programming skills, and country.

**A map of the world

AI-generated content may be incorrect.**

***Figure 10: Choropleth Map – Number of Respondents by Country***

A graph of different colored bars

AI-generated content may be incorrect.A graph of a distribution of salary

AI-generated content may be incorrect.Figure 10 is the interactive map presenting survey response distribution by country. It shows countries with the highest number of developer participants, the United States, India, and Germany leading with the largest figures. The figure supports the global diversity of the dataset and helps in follow-up country-grouped analysis.

***Figure 11: Boxplot – Annual Income Distribution and Bar Chart – Top 10 Programming Languages***

Figure 11, a boxplot showing the distribution of developers' income. As you can observe in the graph, most of them earn less than $150,000, as there is a pronounced right-skewed distribution along with some extreme outliers. From this visualization alone, it seems reasonable to set the median salary as the cut-off for what constitutes "high-income" developers.

The most popular programming languages among programmers are reflected in the bar chart of Figure 11. The top tech stack is occupied by HTML/CSS, JavaScript, and Python, but TypeScript and SQL are also commonly used. This is a representation of programmer proficiency with full-stack technologies and industry demand at the time.A graph of different colored squares

AI-generated content may be incorrect.

Experience of developers in various work categories (full-time, student, and freelance) is contrasted in this boxplot in Figure 12 by income levels. Full-time employees are high earners with high experience. Students and part-time workers, on the other hand, report lower experience and salary. This plot assists in identifying significant correlations between work status, experience, and salary.

***Figure 12: Boxplot – Years of Experience by Employment Type and Income Group***

A graph of a graph

AI-generated content may be incorrect.The scatter plot in Figure 13 indicates the relationship between experience coding and compensation. Experienced developers tend to earn more money. There are some interesting outliers, though, suggesting that experience is not the only determinant of compensation. This lends credence to the suggestion that classification models should use many features.

***Figure 13: Scatter Plot – Years of Experience vs. Annual Income***

A graph of a bar graph

AI-generated content may be incorrect.

The percentage of high- and low-income developers among the most prevalent types of countries (USA, UK, India, Germany, Others) is displayed in this comparison bar graph in Figure 14. It shows that India and the "Others" area have more low-income developers, whereas the USA and Germany have more high-income developers. This graph is simpler to comprehend how income potential is influenced by location.

***Figure 14: Bar Chart – High vs Low Income Developers by Country***

# Cluster Analysis

**3.1 What is Clustering?**

Machine learning models include clustering methods, which involve segmenting the unlabelled data or points into different clusters so that comparable points are in one cluster rather than points that are not comparable with others. In other words, the purpose of the clustering technique is to dissociate groups with similar properties and put them in clusters (Saurav Kaushik, 2024).

Clustering in this project was employed to group developers from the Stack Overflow 2024 Developer Survey into meaningful categories based on their demographic and professional profiles, i.e., income, experience, education, and location. The grouping enables the recognition of hidden patterns, i.e., the identification of what types of developers tend to have more income or which developers share similar characteristics.

**3.2 Data Preparation for Clustering**

The seven selected features were thoroughly checked for the clustering process. Handled missing values in the final section. Then, multi-label fields like "DevType" and "LanguageHaveWorkedWith" were simplified by taking the first mentioned one. Then, we encoded categorical data using pd.get\_dummies() to convert textual data into numeric data. Then, the data was standardized using "StandardScaler" to normalize feature scale before clustering.

* 1. **Finding the Ideal Number of Clusters**

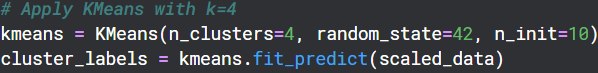
**A graph with blue lines

AI-generated content may be incorrect.**

***Figure 15: Silhouette Score Plot***

With the use of the Silhouette Score method, the optimal value of clusters (k) was discovered. The silhouette score, between -1 and +1 (more is better), determines how well each point lies within its cluster compared to others.

k = 2 to 10 were tried in a loop. The most densely grouped and maximally distinguished clustering in this dataset were four clusters, as was suggested by the score peak at k = 4. From Figure 15, we can see the change in clustering quality with different values of k. The slope after 4 indicates that the k is at its peak when it's in 4. Subsequently, k = 4 was used to apply the KMeans method. The developers were each mapped into one of the four groups according to similarity in their encoded and scaled attributes.



Further preprocessing and optimization of chosen features showed k = 4 to yield the most interpretable and consistent clusters, although previous silhouette analysis on raw or sampled data may suggest optimal k = 9. The subsequent analysis and plots in this report are all performed with k = 4.

* 1. **Cluster Classifications and Visualizations**

The high-dimensional feature space was projected onto two principal components through Principal Component Analysis (PCA) for graphically representing the clusters obtained through KMeans (k = 4).

Each of the developers is plotted as a point in 2D space that has been coloured according to its cluster in the scatter plot, which is shown in Figure 16. Although there are a few intersections, as occurs in social information, the map shows that the clusters are quite different.

This graph shows how the clustering process could put similar developers in the same group according to technical background, education, job, experience, and salary.

A graph with many colored dots

AI-generated content may be incorrect.

***Figure 16: PCA Visualization of Clusters***

The bar plot, which displays the number of developers per cluster, is seen in Figure 17. In comparison, Cluster 2 is the smallest group, which may indicate less experienced developers with intermediate salaries. As we can see from the graph, Cluster 1 comprises a considerable share of responses, suggesting that high-income professionals are well represented in the dataset. The bigger size of Cluster 3 suggests that more respondents are either students or entry-level developers.

A graph with multiple colored bars

AI-generated content may be incorrect.

***Figure 17: Bar Plot: Number of Developers per Cluster***

A brief description of the four clusters produced by the KMeans algorithm is given in the table of Figure 18. Average years of experience, average salary, and a manually derived interpretation of the developer group are used to describe each cluster. This summary fills the information gap between raw cluster IDs and actual usage. For example, Cluster 1 consists of experienced professionals with high incomes; Cluster 0 consists of students or recent graduates with low incomes. To gauge the effectiveness of clustering in aggregating developers that were similar, this table is invaluable

A screenshot of a computer

AI-generated content may be incorrect.

***Figure 18: Cluster Summary Table***

A comparison of different colored bars

AI-generated content may be incorrect.

***Figure 19: Bar Plot: Average Salary and Experience per Cluster***

In Figure 19, we can see that Cluster 0 has the lowest salary and experience, suggesting that it includes students, interns, or early-career developers. If you look at Cluster 1 in the salary bar plot, which shows higher income, and Cluster 1 in the experience bar plot, which shows a median level of experience, it indicates that it could be a group of seniors or full-time professionals. Furthermore, if you look at the Cluster 2 and Cluster 3 in the salary bar plot, it shows almost the same level, but the experience graph differentiates them by a slight difference that we can see in Cluster 2 in the experience graph, which has the third highest experience level, while Cluster 3 shows the highest experience. From the graph, we can conclude that the table from Figure 18 matches the graph.

Cluster 0—Low-income, entry-level, or student developers  
Cluster 1—High-income, experienced professionals

Cluster 2—Moderate-income, early-career developers

Cluster 3—Experienced but underpaid developers

* 1. **Summary**

The KMeans method was utilized in the cluster analysis of this project to group developers into specific groups using seven chosen characteristics: ConvertedCompYearly, YearsCode, Employment, LanguageHaveWorked, Country, EdLevel, and DevType After data preprocessing, encoding, and scaling, the silhouette score method was utilized to conclude that k = 4 was the ideal number of clusters. Principal Component Analysis (PCA) was utilized to conduct dimension reduction so that the clusters are shown in two dimensions to be able to visualize the clustering structure. This demonstrated that, even with some overlap, clusters are prone to separate well along economic and professional characteristics.

Supporting visualizations such as the PCA scatter plot, salary and experience bar charts, and summary tables enhanced cluster interpretability. The integration of all these discoveries enables differentiation across developer segments based on income, experience, and job type — obtaining workforce diversity insights in the technology sector.

The table in Figure 18 summarizes the interpretation for each cluster. The Next Section is the implementation of machine learning and its techniques.

# Machine Learning for Classification and their Implementation

A diagram of a model

AI-generated content may be incorrect.Before the machine learning phase, it's necessary to split the data flow into separate stages to define and know each step in detail. This enables a more structured process for analysing and interpreting the results, which eventually means more precise conclusions and insights. Figure 20 shows the workflow and data preparation.

***Figure 20: Workflow and Data Preparation***

Machine learning classification begins with raw data and undergoes several structured steps before producing a predictive model. The workflow illustrated in the flowchart is an industry-standard method that ensures the reliability, generalizability, and interpretability of the model. First is the original dataset, which, in this case, is the Stack Overflow Developer Survey 2024. It includes over 65,000 developer responses from all around the world, and it covers all the components such as employment, education, experience, salary, languages they worked with, and development type. Next comes the data preparation. This involves cleaning and preprocessing the raw data. Handling the missing values by dropping the NaN value. Simplify multi-label fields by keeping the first value for fields like DevType and LanguageHaveWorkedWith. Encoding the categorical data by using one-hot encoding to convert categories into numerical form. Scaling numerical features by standardizing values like YearsCode and ConvertedCompYearly using StandardScaler. Next is the dataset splitting into

1. Training Set—used to train the model
2. Validation Set—used to fine-tune hyperparameters.
3. Test Set - used for final evaluation and performance metrics.

In this project, we used the "train\_test\_split" method to perform a split with a 70/30 ratio, and cross-validation was handled internally via "GridSearchCV". Multiple classification algorithms were trained using the processed training data. For our coursework, the following classifiers were implemented: logistic regression, decision tree, and k-nearest Neighbor (k-NN). All this model was trained to predict whether a developer is a high-income earner based on selected features. Next is to fine-tune the model; hyperparameter tuning was performed to improve model performance. This involves parameters such as "max\_depth" and "min\_samples\_split" in Decision Tree. Using "n\_neighbors" in k-NN. The tuning was done using the "GridSearchCV" method, which tests different parameter combinations. The next step is to evaluate the model using accuracy, precision, recall, F1 score, and confusion matrix to ensure its performance on the test set. Finally, the best-performing models, including an ensemble voting classifier, were selected as the final predictive model, which now can be used to predict high or low income for future developers based on the survey.

**4.1 Logistic Regression**

Logistic regression is a widely used supervised learning binary classification algorithm. Unlike linear regression, which gives continuous outputs, logistic regression computes the probability that a given input belongs to a certain class — for this project, whether a developer earns a high income (above median) or not.The logistic regression model uses a logistic function (sigmoid) to map the result of a linear equation to an interval between 0 and 1. It is particularly relevant when the job is classified and the result is binary. The decision boundary is set using the threshold (default 0.5) to identify the predicted label.

Why Use a Logistic Regression?

1. It performs well when the target and independent variables are linearly separable
2. It performs well with big, sparse data, so it is a good fit for one-hot encoded features like Country, EdLevel, DevType, etc.
3. Logistic regression provides probabilistic outputs, so it is suitable for model confidence evaluation (e.g., ROC curves).
4. The model is interpretable, i.e., we can look at feature coefficients to observe their impact.

In this project, logistic regression classifies developers as high-income (1) or low-income (0) based on demographic and professional attributes.

The following steps were followed to implement logistic regression in this coursework:

1. Data Preprocessing: One-hot encoding was done through "pd.get\_dummies()" to convert them into numerical features. Standardization of features was done using "StandardScaler" to rescale the input space.
2. Splitting: Data splitting was done through "train\_test\_split" (70% training, 30% testing).
3. Model Training: Logistic regression was done through Scikit-learn's "Logistic Regression" class. We included "max\_iter = 1000" to achieve convergence, given the high-dimensional feature space.

A chart with yellow and purple squares

AI-generated content may be incorrect.Logistic regression confusion matrix from Figure 20 indicates good classification performance. The model classified correctly 3451 low-income developers and 3455 high-income developers out of the total 7031 test samples. Only 53 low-income individuals were classified as high-income (false positives), and 72 high-income individuals were classified as low-income (false negatives).

A graph of different colored squares

AI-generated content may be incorrect.These results show just how accurate and balanced the model is for both classes. Such performance is particularly crucial in practical applications where underestimation as well as overestimation of income can generate biased conclusions. The minimal number of misclassifications further depict the robustness of logistic regression on this dataset.

***Figure 21: Logistics Regression – Confusion Matrix***

***Figure 22: Accuracy Bar Comparison***

As indicated by the bar chart showing the accuracy comparison, logistic regression achieved a high 98% accuracy. While slightly less than the ensemble Voting Classifier (99%) and Decision Tree (100%), it performed far better than the k-Nearest Neighbors classifier (78%). This confirms that logistic regression is highly effective for classifying high-income developers from high-dimensional, encoded survey data.

**4.2 Decision Tree**

Decision Tree is a supervised learning, non-parametric algorithm that is used for regression and classification alike. The process divides the data set into subsets based on information gain maximization, which ultimately builds a tree-like hierarchy of decision rules.

In classification problems, the decision tree repeatedly selects that feature which is responsible for maximizing the reduction in entropy (or impurity) using criteria such as Gini Index or Entropy. The tree splits until it reaches the stopping points — e.g., a maximum depth or pure leaf nodes.

A yellow and purple squares with numbers

AI-generated content may be incorrect.

Why Use a Decision Tree?

1. Interpretability: Decision trees produce human-readable rules — you can literally "see" how predictions are being made.
2. No feature scaling required: Unlike some other algorithms (e.g., k-NN or Logistic Regression), decision trees are not concerned with the scale of features.
3. It can handle categorical and numerical features equally well.
4. Good at modeling non-linear relationships.

***Figure 23: Decision Tree – Confusion Matrix***

A diagram of a algorithm

AI-generated content may be incorrect.The following steps were followed to implement Decision Tree in this coursework:

1. Data Preparation: Used the same one-hot encoded data with 7 selected features.

Target variable: highIncome (1 = above median income, 0 = below).

1. Model Training: Performed with Scikit-learn's DecisionTreeClassifier.

Parameters used: criterion='entropy' (for Information Gain)

max\_depth=4 (avoids overfitting, makes the tree interpretable)

random\_state=1 (reproducibility)

The confusion matrix for the Decision Tree model in Figure 23, shows an optimal classification performance of 100% on the test set. The correct classification happened for all 3504 low-income developers and for all 3527 high-income developers, which is to say that there were neither false positives nor false negatives. This can be explained in terms of the structure of a decision tree that depends exclusively on the normalized salary (ConvertedCompYearly) as a splitting rule. The choice of a decision boundary of -0.106 well delineates the above- and below-median income earners among the developers. The leaf nodes are pure and optimal, depicting all high- or all low-income developers. This enhances the fact that this attribute has good prediction capabilities for this classification task.

***Figure 24: Decision Tree for Salary and Experience***

* 1. **k-Nearest Neighbors (k-NN)**

A chart of a confused matrix

AI-generated content may be incorrect.

k-Nearest Neighbors (k-NN) is a straightforward, instance-based learning algorithm for classification and regression. Unlike other models that "learn" a function from the data during training, k-NN predicts by comparing new input samples to previously observed training samples.

In classification, the algorithm computes the distance between the new data point and all training points, finds the k nearest points (neighbors) and assigns the class based on the majority vote among those k neighbors.

Why Use a k-Nearest Neighbors?

1. Simplicity: Easy to use and apply.

***Figure 25: k-NN – Confusion Matrix***

1. No training phase: It keeps the data set in memory and determines at prediction time.
2. A graph with different colored bars

   AI-generated content may be incorrect.Non-linear modeling: k-NN can learn non-linear decision boundaries.

But it does have a couple of significant drawbacks such as sensitivity to feature scaling (so standardization is necessary), Computationally expensive on large data sets and Performance degrades with high-dimensional sparse data (such as one-hot encoded features)

The following steps were followed to implement k-NN in this coursework:

***Figure 26: Precision, Recall, and F1 Score Comparison***

* Model Training: Used KNeighborsClassified from Scikit-learn.
* Parameter: n\_neighbors=5
* Trained on training data, predicted on test data.

This k-Nearest Neighbors confusion matrix in Figure 25 reveals quite a poor accuracy in classifying work samples. Of the total samples tested, the model could correctly identify 2823 high-income developers and 2669 low-income developers. However, the model retrieved 835 low-income A yellow and purple squares with numbers

AI-generated content may be incorrect.developers as high income and 704 high-income developers as low income. In summary, the high misclassification rates resulted in an overall accuracy of only 78%, making k-NN the absolute weakest performer among all classifiers. Dependency on distance metrics and the absence of a learning phase make it an inappropriate approach for the high-dimensional, one-hot encoded dataset. Further verification has been done by the bar chart of model accuracy, in which k-NN again performed very poorly against all the other models.

* 1. **Ensemble Method – Voting Classifier**

***Figure 27: Voting Classifier– Confusion Matrix***

Voting Classifier is an ensemble technique for combining diverse base models in the search for improved predictive performance. This coursework uses hard voting to combine Logistic Regression, Decision Tree, and k-Nearest Neighbors. The predicted class is determined by the highest number of votes from individual models.

The ensemble exploits the strengths of the models:

* Logistic Regression provides generalization and decision boundaries.
* Decision Tree captures non-linear relationships and hierarchical splits.
* k-NN contributes to local pattern recognition, despite being weak on its own.

Together, the Voting Classifier is stronger and compensates for bias and variance.

In the confusion matrix in Figure 27, one can easily see a strong model performance, with 3478 high-income developers and 3462 low-income developers classified correctly. Only 42 false positives and 49 false negatives occurred, which resulted in a very low misclassification rate and strong generalized ability across income classes.

The Precision–Recall–F1 Score plot in Figure 26, shows the general Voting Classifier to perform almost perfectly:

* Precision and Recall are very close to 1.0, which means the complete and accurate classification of both income classes.
* the F1 Score, the harmonic mean of precision and recall, confirms that both measures are strongly balanced.

Thus, the Voting Classifier is the most balanced and therefore reliable model, combining the advantages of individual classifiers while mitigating their disadvantages as far as other models are concerned.

# Evaluation Machine Learning Models

A host of evaluation metrics - namely, accuracy, confusion matrix, precision-recall-F1 score-ROC curves- were used to assess the performance of the implemented models. Each model was tested on identical splits of datasets through the identical preprocessing pipeline for fair comparison.

The bar chart from Figure 22, shows the model accuracy comparison. Logistic Regression showed the accuracy of 98%, Decision tree showing the accuracy of 100%, k-NN showing the accuracy of 78% and Voting Classifier shows accuracy of 99%. From the record, we can say that:

* The Decision Tree achieves perfect accuracy. However, this could be due to overlifting as it only relies on the salary feature.
* k-NN performs the worst, as it likely is due to the high-dimensional and sparse feature space.
* The Voting Classifier is the most reliable and generalizable of the models, and it just edges out Logistic Regression.

Next is to compare the confusion matrix graph. All the graphs are implemented in their section.

Logistic Regression: 3451 low-income develops along with 3455 correctly classified high-income developers. Total: 125 misclassifications. A fair performance was slightly lower on the false-positive end.

Decision Tree: Perfect classification of both income groups. It is just too impressively simple (on one feature) for generalisation.

k-Nearest Neighbours: Very high error rate: 835 false positives, 704 false negatives. Very low ability to generalize because of their poor distance metrics in high-dimensional space.

Voting classifier did just that: it correctly predicted 3,462 low-income and 3,478 high-income developers. Only 91 misclassifications occurred. Strong generalization by being able to combine strengths of individual models.

A table with numbers and text

AI-generated content may be incorrect.Below is the table showing the observation of Precision, Recall and F1 Score.

***Figure 28: Table for Precision, Recall and F1 Score***

From the table we can conclude that:

* Overall, the Voting Classifier displays the best performance across all metrics, with relatively high scores.
* The k-NN method, on the other hand, has consistently demonstrated that it is worst with most scores, especially precision.
* Logistic Regression performs quite strongly but undermatches the ensemble somewhat in the metrics.
* A graph of a curve

  AI-generated content may be incorrect.Decision Tree’s exceptional results may have been influenced by overfitting to the training set.

***Figure 29: ROC Curve for All Models***

The Receiver Operating Characteristic (ROC) curve is a powerful tool for visualizing classification model performance. This allows one to plot the True Positive Rate (TPR) against False Positive Rate (FPR) at different threshold levels. The diagonal dashed line represents a random guess classifier (AUC = 0.5). The nearer a model’s ROC curve is to the top-left corner, the better it is at differentiating between the classes.

What does Figure 29 show?

* Logistic Regression (Blue Line):

AUC = 1.00. The model is said to perfectly separate the high-income class from the low-income class. There was no need for trade-offs; both false positive and negative rates were minimized.

* Decision Tree (Orange Line):

AUC = 1.00. Indicates perfect classification, which may imply overfitting since it depends mainly on one dominant feature (ConvertedCompYearly).

* k-NN (Green Line):

AUC = 0.85. It has a much less optimal curve shape. Shows high false-positive and false-negative rates across the thresholds. It has difficulty separating boundaries and works poorly in high-dimensional spaces.

* Voting Classifier (Red Line):

AUC = 0.99. Very close to perfect. Slightly below Decision tree and Logistic Regression but infinitely more robust. For generalization vs. accuracy trade-off, this is the best classifier.

# Discussions and Conclusions

Profoundly gratifying and mostly practical, this project allowed me to apply the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology to a real-world problem: predicting high-income developers using survey data collected in the 2024 Stack Overflow Developer Survey. By adhering to every component in the CRISP-DM cycle from business understanding, data understanding, data preparation, modeling, evaluation, and finally deployment, I have thereby consolidated both my practical and conceptual data science knowledge.

The CRISP-DM way enabled me to see how important it would have been to have a structured workflow for a data project. Some clear objectives were set for fields: whether a developer earns above-average salary or not; this business objective was then translated into a data-science-oriented goal through a series of systematic investigations of the dataset. It involved an exploratory data analysis, data understanding, and the application of different techniques to identify meaningful patterns and relationships involving statistical summaries as well as several forms of visualizations like box plots, scatter plots, choropleth maps, and bar charts. This enabled me to identify the more particular variables strongly associated with the developer's compensation, namely Employment, Education Level, and Years of Coding Experience.

Data cleaning and preprocessing are the most enlightening for me. Missing value treatment, normalizing numerical data, encoding categorical variables, and certain transformations on multi-label columns needed to be very careful and added tremendously to furthering my understanding of data quality and feature engineering. I came to see the imperative nature of cleaning and preprocessing data before anything else, such as applying machine learning models.

I learned to apply the unsupervised techniques below when building an analysis of clustering with k-Means and visualizing the results using PCA. It brought forth the patterns and subgroups in the data, such as clusters of full-time professionals, students, and freelancers. The elbow method and silhouette scores were employed to govern the optimum count of clusters, and I learned to interpret the characteristics of each cluster based on the average income and experience levels.

The classification section itself was the most profound and technical part of the coursework. I implemented three primary classification algorithms: Logistic Regression, Decision Tree, and K-Nearest Neighbors (K-NN). I also defined an Ensemble Voting Classifier that blended the strengths of the individual models for reliable predictions. The evaluation of these models using parameters such as accuracy, confusion matrices, precision, recall, F1 score, and ROC-AUC curves gave me confidence in ultimately choosing the best model. The Decision Tree has made these perfect scores, but it was simple and based on one feature, unlike the k-NN that did much poorly in all metrics compared to the others. On the other hand, the Voting Classifier made it to a very high score in all metrics, which made it the most balanced and robust model.

Writing the report along with technical implementation has proven quite enlightening indeed, for it brought me to very critical reflection at each stage, justifying decisions and practicing clear communication of findings within data science, which, in my mind, is just as important as technical know-how. Being oriented towards CRISP-DM would allow me to know the weight of methodological rigor and transparency in presenting analytic work. This project has added to my confidence in analysing real-life data as well as solving complex problems using machine learning algorithms. It also showed the importance of ensemble learning techniques such as the Voting Classifier, which was found to be better than any of the individual models and has proven to have very strong generalization capabilities. In closing, not only did the project open a stronger base for my technical and analytic skills, but it showed me how structured frameworks like CRISP-DM could lead to clear, sound, and interpretable data science outcome.

# References

1. Shawn Chumber, 2023. *The CRISP-DM Process: A Comprehensive Guide* [online]. Google: Medium. Available at: <https://medium.com/@shawn.chumbar/the-crisp-dm-process-a-comprehensive-guide-4d893aecb151> [Accessed 16 March 2025].
2. Henry Ha, 2024. *Data Science Project Lifecycle-An Overview of CRISP-DM and the Formulation of Data Science Problems* [online]. Google: GoPenAi. Available at: <https://blog.gopenai.com/data-science-project-lifecycle-an-overview-of-crisp-dm-and-the-formulation-of-data-science-problems-c00a88ca447e> [Accessed 16 March 2025].
3. Richard O’Hara, Lisa S, Haylon, Douglas M. Boyle, 2023. *A Data Analytics Mindset with CRISP-DM* [online]. Google: StrategicFinance. Available at: <https://www.sfmagazine.com/articles/2023/february/a-data-analytics-mindset-with-crisp-dm> [Accessed 17 March 2025].
4. Figure 1: Richard O’Hara, Lisa S, Haylon, Douglas M. Boyle, 2023. *CRISP-DM Diagram.* Available at: <https://www.sfmagazine.com/articles/2023/february/a-data-analytics-mindset-with-crisp-dm>
5. Ana Azevedo, Manuel Filipe Santos, 2008. *KDD, SEMMA AND CRISP-DM: A PARALLEL OVERVIEW* [online]. Google: ResearchGate. Available at: <https://www.researchgate.net/publication/220969845_KDD_semma_and_CRISP-DM_A_parallel_overview> [Accessed 17 March 2025]
6. Fayyad, U., Piatetsky-Shapiro, G. and Smyth, P. (1996). “From Data Mining to Knowledge Discovery in Databases”, *AI Magazine*, 17(3), p. 37. doi: 10.1609/aimag.v17i3.1230.
7. Sameer, 2024. *Types of Plots: Visualization from Concept to Code*[online]. Google: AnalyticsVidhya. Available at: <https://www.analyticsvidhya.com/blog/2021/12/12-data-plot-types-for-visualization/>
8. Fleckenstein, M., Fellows, L. (2018). Big Data and Data Management. In: Modern Data Strategy. Springer, Cham. <https://doi.org/10.1007/978-3-319-68993-7_2>
9. Saurav kaaushik, 2024. *Clustering | Different Methods, and Applications*[online]. Google: AnalyticsVidhya. Available at: <https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/>
10. Comal.tech, 2020. *Worlflow and Data Preparation* [online] Available at: <https://www.theobjects.com/dragonfly/dfhelp/2020-1/Content/Artificial%20Intelligence/Deep%20Learning%20Tool/Workflow%20and%20Data%20Preparation.htm> [Accessed 01 April 2025]

# Appendix

Figure 1: Years of Coding Experiece

A graph of a number of years

AI-generated content may be incorrect.

Figure 2: Full page spread of Years of Coding Experience

A graph of numbers and lines

AI-generated content may be incorrect.

Table 1: First 5 rows of selected features

