Machine Learning for Data Analysis

Report

Terrence Oppon | Foundations of AI and Machine Learning | N1004932

# Contents Page

**Section 1 – Introduction to Report**………………….…………………………………………………………………………2

Introduction

**Section 2- Data Understanding, Data Pre-processing, Exploratory Data Analysis**…………………….5

Background Information of data

Exploratory Data Analysis

**Section 3- Cluster Analysis**………………………………………………………………………………………………………13

Subset analysis

Conclusion

**Section 4- Machine Learning for Classification and their Implementation**………………………………17

Decision Tree

k-Nearest Neighbour

Logic Regression

Ensemble Learning

**Section 5- Evaluation of Machine Learning Models** …………………………………………………………………23

Accuracy

Confusion Matrices

Decision Tree classifier

k-NN classifier

Logic regression classifier

ROC Curve

**Section 6- Conclusion on Machine Learning Coursework**…………………………………………………………27

Results

# Section 1- Introduction

In this coursework, I will be analysing and implementing the data from "the 2021 freeCodeCamp New Coder Survey". This survey shows us all significant insight into how thousands of people all around the world are learning to code, and why they are deciding to start coding. As more than 18,000 people have started to pick up coding and respond to this survey, this report will show us the sudden rise in people learning how to code. We’ll also be implementing a cluster analysis to understand all of the sudden new coders and we will also be looking to see if a new coders income these people are making.

The Cross-Industry Standard Process for Data Mining” (otherwise known as CRISP-DM) is a popular framework for data mining and analytics initiatives. First presented in 1996 by a group of European data mining specialists, the technique consists of six primary phases:

1. Business Understanding,
2. Data Understanding,
3. Data Preparation
4. Modelling
5. Evaluation
6. Deployment

It offers a structured strategy for data mining and analytics initiatives.

1. **Business Understanding**

The first phase of CRISP-DM is about understanding the business. Understanding the project objectives and identifying the issue that needs to be solved is part of the business understanding phase. This is helpful, as it helps business processes be defined and helps the plan to proceed with limited complications.

1. **Data Understanding**

During the data understanding phase, all the data is located and gathered. When all of the data has been found, the data is examined to learn more about the data's quality and any potential complications.

1. **Data Preparation**

The third phase of data preparation involves cleaning and processing the data. This phase is also important, as in this phase, choosing the features that will be used in the modelling step are all part of the data preparation phase.

1. **Modelling**

The modelling phase is used for choosing and creating the best model(s) to solve the current problems. In this phase, there are also test created to check the validity of the models that are going to be used. It’s good to assess these models to make sure that all models that are going to be used, are useful for the business initiatives.

1. **Evaluation**

In the development phase, the model developed in the previous stage is evaluated in this phase to check if they carried out their job.

1. **Deployment**

The development stage is the final stage and is key, as its important to creating and implementing the models.

I will be implementing CRISP-DM into my coursework by CRISP-DM will be used in my project because it is an appropriate methodology for the classification issue being looked at. CRISP-structure DM's of emphasising complete data understanding before the preparation and modelling stages may be helpful

# Section 2- Data Understanding, Data Pre-processing, Exploratory Data Analysis Exploratory Data Analysis (EDA)

Background Information of data

The data that we are going to be analysing was collected by the 2021 new coder survey. Around 18,000 people participated in the survey. They were asked 49 different questions on GitHub and Kaggle to find out the sudden uprise in new coders. The main purpose of collecting this data is to find out why a lot of people are starting to learn how to code, how they are learning, and where people are learning the most are from. This survey provides us with data ranging from continent, race (in certain counties), age and other sorts.

Key Data Attributes

|  |  |  |
| --- | --- | --- |
| Attribute Number | Question | Data type |
| 1 | What is your biggest reason for learning to code | object |
| 7 | About how many hours do you spend learning each week | Float |
| 9 | how much money have you spent on learning to code so far (US Dollars) | float |
| 14 | Which career path are you interested in | Object |
| 23 | How old are you? | Float |
| 26 | Which part of the world do you live in? | object |
| 32 | What is the highest degree or level of school you have completed? | object |

The table above shows us which of the important attributes that we’ll be diving into as they show what we are looking for and how these questions match up to this report’s tasks. But to carry on with this analysis, we need to make sure we cleanse all the data.

First, I created a table with the key attributes from the table above.

Graphical user interface, text, application

Description automatically generated

## 

There are multiple values that have ‘NaN’ in the data. This means that they are missing values that we must find to continue. I created another table to find the values of the missing data.

Graphical user interface, application

Description automatically generated

Now that we have the data for the missing values, we can now calculate the percentage of the responded developers that answers each question. The results are as followed:

* 95.08% of respondents answered the question in column Area.
* 99.26% of respondents answered the question in column Reason.
* 84.86% of respondents answered the question in column Income.
* 91.60% of respondents answered the question in column Hours.
* 94.92% of respondents answered the question in column Path.
* 94.36% of respondents answered the question in column Age.
* 89.12% of respondents answered the question in column Spend.
* 95.95% of respondents answered the question in column Interest.

The missing values will heavily affect our progress in the future so, we have to drop the missing values from the dataset. Even though this will lead to a reduction of columns and a loss of data, we can now continue on with the project if we handle the values correctly.

Now I will show you the updated data frame.

Graphical user interface, text, application, email

Description automatically generated

## Exploratory Data Analysis (EDA)

Exploratory data analysis or often referred to as EDA is the examination of data through the exploration to find any patterns or anomalies is known as exploratory data analysis. The following section will concentrate on looking at the crucial features that have been chosen to fully comprehend the data.

1. **Biggest reason to code**

To start off with the first question we analysed, the main and general question, which is why people wanted to code in the first place. To check this out. We simply conducted a pie plot with all the different categories to see why people have started to learn how to code. Since there are so many answers, we simplified it to show the most popular ones. We chose to display the top 3 results, as they are the, as we see, the most important ones.

Graphical user interface, application

Description automatically generated

From the results, we can see that the most popular option is because people want to change their career. This information is useful, as it can help us see the new developers and why there are so many. In this pie graph, they dominate by having nearly 60% of the vote. This could be due to the rise in income that new developers get, that they want to change their career. We can link this info to the data from the age column, which we will look at later.

We can also create a bar plot to see around about the numbers of people that have the same opinion.

Graphical user interface, application

Description automatically generated

This time, we have shown the top 5 results and from the numbers here, we can see that around 3500 people said that they would like to change careers. That is way more than the next highest which is to start their first career, which had around 1400 responses to it.

Finally, like we mentioned earlier, it would be interesting to compare this to the ages of the respondents, to see if the age of the people corresponds with the reasons to code.

Graphical user interface, application

Description automatically generated

As we can see here, the option to start you first career, has the youngest respondent and in general, has the lowest highest age compared to the rest. This makes sense, as young people are always looking for a new career they can look into. The to change career option has a range from around young teenagers all the way to around the late 70’s which is surprising to say the least

1. **Hours per week spend coding.**

The next Option we will look at is the amount of time spent coding. There was 16.6k valid results with only 1523 missing. This topic is a very interesting point, as there could be outliers to see if people are way below or above the norm. To see this, first of all, we’ll be using a histogram, to compare the hours worked on coding, and the age of the people who took the survey, as I feel like this is a good comparison point for this section.

Graphical user interface, application

Description automatically generated

Looking at the data, we can mainly see that as the hours increase, the amount of people in general start to increase. This shows that the older people aren’t really spending a huge amount of time learning to code compared to the younger people. This is interesting as in the previous section about reasons to code, it was usually the older people who wanted to learn.

Next, we’ll be looking at a scatter and a box plot to see if we will have any. If there are any outrageous results, we will clean the data and remove any insane results, by giving a valid range of hours worked per week.

Chart, scatter chart

Description automatically generatedGraphical user interface

Description automatically generated

As we can see, there are a few outliers, as some people said they work for 140 hours in a week, which wouldn’t be realistic. To get a realistic average, we decided to give a range from 0-80 instead. This also gives us a more precise and better look at the data.

Chart

Description automatically generated

Graphical user interface, application

Description automatically generated

We also did an age range from once again 0-21

Graphical user interface, application

Description automatically generated

We also made a pie chart to make it easier by looking at the percentages. And we can see that rather surprisingly, 2-5 hours was the most popular. Finally, we made a histogram to compare the income level and the hours spent coding to get a direct comparison of the effect of income level from hours put in.

Graphical user interface, application, Word

Description automatically generated

1. **Money spent on learning to code.**

The next area was going to look at Is the amount of money spent learning to code. I have decided to categorise this by where people are from as this will give us an accurate representation of each area. To do this, I created a crosstab to directly compare both categories. at What the results tell us is that most people in fact don’t spend anything learning to code. For example in East Asia and Pacific, most people answered “0” when it came to spending money on coding. Compared to North America, even though 0 was the most common result, they were the most varied with people spending 99 pounds and so on.

Graphical user interface, text, application

Description automatically generated

To further prove this, we can split this into Head and Tail, the head has majority of its numbers in the ‘0.0’ row, which shows that people aren’t spending money to learn. However, if we take a Graphical user interface, application

Description automatically generatedcloser look at the bottom have (tail section) it shows that only a couple continents have people paying, with North America having the most, Graphical user interface, text, application

Description automatically generatedand regions like Asia, having none.

1. **career Interest**

Now we’re going to be looking at career path and how that has an effect on new developers. To view this, we simply made a pie plot, to view all the possible outcomes and its percentages. Once again due to many responses, we simplified it to show the most popular ones. We found multiple results but the most popular ones was the full-stack web developer. This could be down to a multitude of reasons like money and all sorts.

Graphical user interface, application, Word

Description automatically generated

We also made a bar plot so we can see the 5 most popular career paths.

Graphical user interface, application

Description automatically generated

We get a closer look at the numbers in this view. The results in this section makes a lot of sense as in the very first section, Reasons to learn coding, a lot of people wanted to start a business, so web development can become a big part of that.

1. **Age**

Next, we’re going to be looking at age, and how that varies when it comes to people and learning to code. There were about 8974 valid answers to this question since a lot of people answer. This could be because people were either too young to remember the age they started, or they didn’t want to give their age. The data type for the data collected was a float, which lead to some results not giving us whole numbers. The range of the answers was between 6-120 however, the highest age that was present was 69 years old. We can create a scatter plot to further analyse this data and see for any anomalies.

A picture containing graphical user interface

Description automatically generated(With Missing Values Removed)

We can see a couple outliers here, as one person has even claimed to have started coding from after the Age of 100. To eliminate these anomalies, we decided to put a range on the age from 0-70. Even though this will get rid of some results, it’ll make our findings more accurate.

Chart, box and whisker chart

Description automatically generated(Boxplot with restrictions)

Another way we can analyse the age and the cross-comparison of developers is to compare the ages of the developers and the amount of income they have received. The way we are going to analyse this is by doing a scatter plot, to see the relationship between the age of these developers and the time they spend coding.

Graphical user interface, text, application

Description automatically generated

As we can see, most people spend between 0-60 are spread out evenly between all income values. The older people from 60+ are all not nesicarrily earning the highest about, as even the oldest is earning from 35k-39k.

A picture containing table

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

Chart, scatter chart

Description automatically generated

One impact this could have been the salaries they are all making. This is because the older you are, the more time you could have as you’d want to get a job coding/ focus on coding, compared to younger people/ students who have to spend time at school and university. Vice-versa, adults have jobs and could have children to look after as they have more responsibilities.

Of course, there would be external factors that would play into this, such as where they work, the country they are from, hey have how much spare time and so on however, this correlation is still a valuable insight. The reason for this comparison is because depending on the time you have to improve your skills, you can earn a higher income, so this looks at the impact of age and progress.

Finally, we conducted a Pie chart to see the Ages of our new developers. From what It seems, just over 50% of the new developers are 20-30 while there’s just 0.8% aged 60-70.

Graphical user interface, chart, application, pie chart

Description automatically generated

1. **Where do you live?**

For the nation attribute, it would be interesting to see and compare where everyone is from. We managed to gather the majority of where people are from, which included the respondents' country of origin, notably the continent. We can work out the percentages of developers and where they are from by creating a pie plot and a bar plot. This will show us where mainly where the new developers are from.

Graphical user interface, application

Description automatically generated

A picture containing graphical user interface

Description automatically generated

As we can see judging by each continent, we can see that North America has the newest developers compared to the other continents. North America has a total of 37.4% and the next highest is central Europe and Central Asia with just 22.3%. Middle East and North Africa have the least new developers with just 3.1%. Comparing it to the Income.

This had me intrigued to look at two more things. Where you are from and the amount of hours you spend coding First of all in the bar plot, we can see that Europe and central Asia spend the most time coding followed by North America. East Asia and pacific have the least by around a difference of 70 hours, to the nearest whole number.

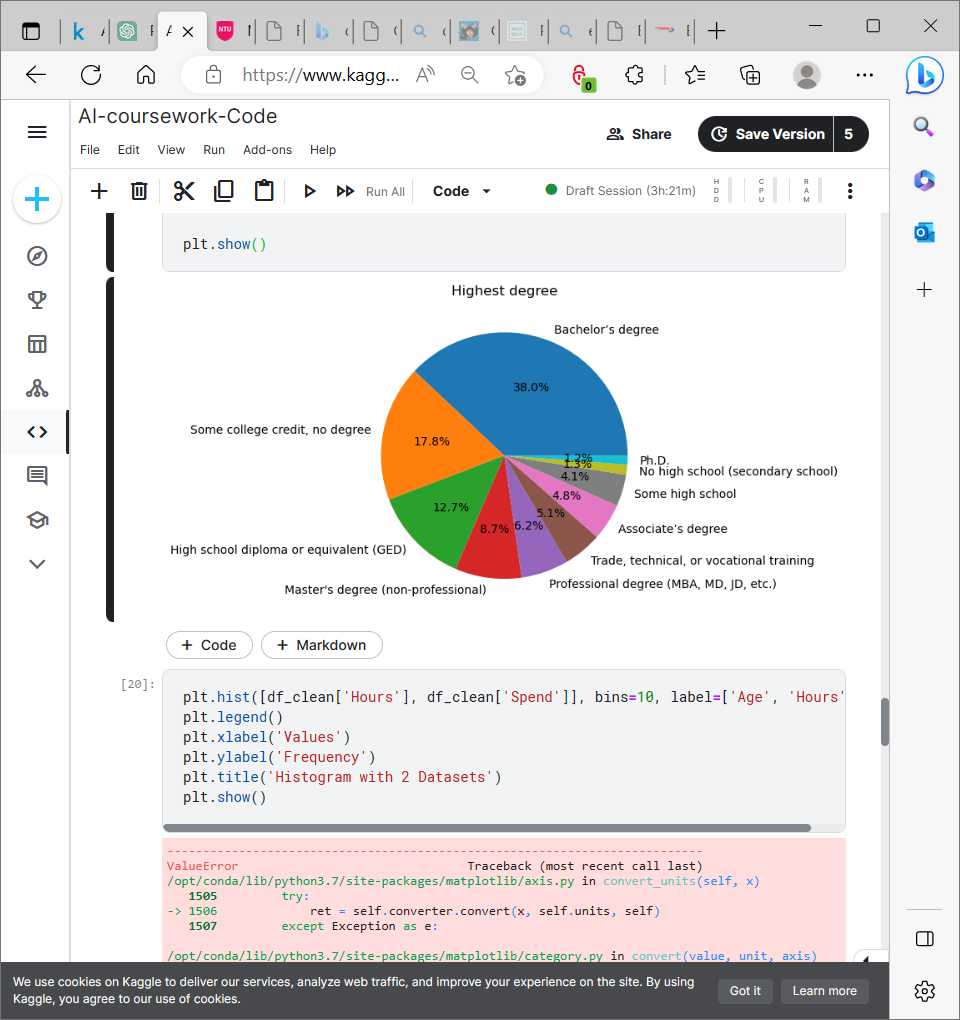
Graphical user interface

Description automatically generated

I also compared this to the amount of money spent. From the crosstab from before, we had interesting info like, only a couple continents have people paying, with North America having the most and regions like Asia having none, but now, I want to put it in a bar plot for a better and easier viewing and comparison.

1. **Highest degree or level of school**

Finally, the last factor we will be looking at is the highest degree a new developer has achieved. Most new developers will have some kind of degree or accomplishment. First, we can look at a pie chart and see the percentages of all the new developers and their highest degree achievement.



We can see that by the percentages, we can see that the bachelor's degree option was chosen most of the time and has a percent of 38%. This is a massive percent compared to the other options. The Ph.D. option was chosen the least frequently with only 1.2% of people choosing it, with the No High school option just ahead, with1.3%.

We can also look at a bar chart to see the most common degree.

A picture containing graphical user interface

Description automatically generated

From the data above, we can see that the bachelor's degree option was chosen most of the time, with above 3000 votes. On the other hand. the Ph.D. option was chosen the least frequently. Another interesting point is that the Some collage credit/no degree option was also one of the more popular votes, getting the second highest votes.

Comparing to see If education level has an impact on Income made. To do this, I compared the income level to the degree achieved and found compared the rates. Graphical user interface, application

Description automatically generated

# Section 3- Cluster Analysis

Next Up we’re going to be doing a cluster analysis of the data. Cluster analysis refers to gathering data/data sets and groups them into clusters. This is so that we can identify patterns in a data.

Cluster analysis is extremely useful, as we can see the similarity of the objects within the cluster is maximized and clusters are also useful for minimizing the similarity of clusters.

One example and method of clustering data is K-means clustering, which we’ll be looking at today. K-means is a straightforward and effective algorithm for finding clusters in data. K-means clustering uses the mean values of the data points in each cluster. Using the mean values, K-means clustering places each data point into one of k groups.

Another cluster method is hierarchical clustering. In hierarchical clustering, clusters are organised into a dendrogram-like structure that resembles a tree. This time, the data set will be split up into two subsets based on the income amount of the developer. The major properties of the subsets will be subjected to cluster analysis techniques to find patterns.

In this project, we’ll be using k-mean clustering due to the output data being similar. K-mean clusters have many parameter settings that effect the results. The ones that we used were:

1. Number of clusters: n\_cluster = 5. Three clusters were chosen as the number of clusters since it produced results that were easier to examine than other numbers.
2. Graphical user interface, application

   Description automatically generatedRange: model, k=(2,10). This code instructs the visualizer to calculate the distortion scores for values of K ranging from 2 to 10 by providing k=(2,10).

Chart, line chart

Description automatically generated

## Subset analysis

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedNext, we are going to get the 5 different clusters and compare then to each other. Before performing cluster analysis, we must change the "Education Path" and "Area" attributes, as they are good to compare them to each other and they are very similar categorical data. This is because they are now in categorical format and cannot be used. Because there is a ranking order among the values that we can find to a particular value, we can convert the value "Education Path" by using ordinal encoding. On the other hand, we are unable to apply this encoding technique to the value "Area" since we are unable to establish a connection between the values.

As we can from the count plot data, All the clusters has North America as having has the most new developers, as Cluster 3 has the highest proportion of developers from North America, with cluster 0 not being too far off and with cluster 2 having the lowest proportion, as developers from other Continents is almost as high as north America.

Also looking more at the data with the education path, cluster 0, 3 and 4 seems to have the highest proportion of developers with a Professional degree compared to the rest of the qualifications within its own cluster, and with cluster 3 having a professional and master’s degree being the same. Due to how close cluster 2 is, especially with cluster 2 having the highest proportion of a professional and master’s degree compared to the others, the data, developers in cluster 2 often have higher levels of education than those in cluster 0, 3 and 4, which tends to be lower.

We also conducted a scatter plot, with a colour-coded key to show each cluster, so we can compare them to each other. We compared the Hours worked per week by new developer and the age of each developer.

We also show the low and high income cluster versions to compare the two.

## Low Income Subset

Application

Description automatically generated with low confidenceScatter plot for age and hours per week

Starting with the cluster analysis for the low income subsets of developers, we may perform a cluster analysis. Five unique clusters that are simple to distinguish have been produced by the clustering procedure. This means that the dataset has been separated into five groups. We can see that people who work high hours (who have been bracketed in cluster 4 for this subset) are mostly between the ages of 20 and 40. Cluster 4 work for 55 hours per week +, which shows that they are the highest workers. Cluster 2 is interesting as there are all aged from around 37-70, and all work from either 0-25 hour, which is interesting. Cluster 0 and 3 are all around the same age, from the teens to late 30’s, however, cluster 3 average more hours than cluster 0. Each of which is identified by a number between 0 and 4. When contrasting the cluster results with the count plots, these labels will be useful as we can easily tell which cluster is which.

## High Income

Graphical user interface, chart, application

Description automatically generated

Now with the cluster analysis for the subsets of developers with high incomes, we perform a cluster analysis. We can see that people who work high hours (who have been bracketed in cluster 3) are mostly between the ages of 20 and 60. Cluster 3 is from 35 hours per week +, which shows that they are the highest workers. Cluster 4 is interesting as there are all aged from around 45+, and all work from either 0-25 hour with little working more. Cluster 0 and 2 all work the same hours, which is <20 hours per week, however, cluster 0 are younger than cluster 2. Each of which is identified by a number between 0 and 4.

## Conclusion reesults

To conclude the low cluster:

1. Cluster 0: In cluster 0, Not many hours spent coding per week, older developers, Most developers with a bachelor’s degree, Most developers living in North America
2. Cluster 1: In cluster 1, decent hours spent coding per week, developers aged 20-40,
3. Cluster 2: In cluster 2, Not many hours spent coding per week.
4. Cluster 3: In cluster 3, Not much hours spent coding per week, younger developers.
5. Cluster 4: In cluster 4, Not many hours spent coding per week, developers cover all ages developers,

To conclude the high cluster:

1. Cluster 0: In cluster 0, Not many hours spent coding per week, youngest developers, Most developers with a bachelor’s degree.
2. Cluster 1: In cluster 1, decent hours spent coding per week, developers aged 20-40.
3. Most developers with a bachelor’s degree, Most developers living in North America
4. Cluster 2: In cluster 2, Not many hours spent coding per week, young developers, Most developers with a bachelor’s degree, Most developers living in North America however, they have the lowest proportion, as developers from other Continents is almost as high as north America.
5. Cluster 3: In cluster 3, Most hours spent coding per week, younger developers, most developers with a bachelor’s degree, highest amount of developers in North America Most developers living in North America
6. Cluster 4: In cluster 4, Most coding per week, oldest developers, Most developers with a bachelor’s degree, Most developers living in North America

The clusters in the high-income based off the scatter plot, compared with the low-income, bracket tend to generally have:

• young developers

• not a lot of hours worked per week

• More developers from wealthier continents with Resources like North America and Europe

• More developers with a bachelor’s degree

## Section 4- Machine Learning for Classification and their Implementation

Machine Learning Workflow Chart

The workflow of machine learning for classification has many different steps to it. The first one is data collecting. This is where we are gathering the necessary information for the task. All the information and data has already been implemented in the previous sections. After that, it’s all about splitting the data. dividing the data into two or more sets—usually training, validation, and test—is a common practise. We’ll be splitting 70% into training and the 30% of the rest into the test. The validation set is used to fine-tune hyperparameters, the testing set is used to assess the model's performance, and the training set is used to train the machine learning model. After, we’ll have to choose a model. The metrics we choose will determine which machine learning algorithm is best for the task at hand. We will need to employ a model to illustrate that since we are determining and forecasting the correctness of the data. We can deploy it once our models have been evaluated and have a high level of accuracy.

I will now show an example of what the machine learning workflow chart could look like.

Chart

Description automatically generated

In my coursework, I chose 3 different classification models to use. To make sure I can get the most accurate results with the fastest time taken to get these results, I decided to choose the decision tree model, the k-Nearest Neighbour model and the logic regression model.

The dataset that I have had be split up into a 70% 30%. The 70% is representing the training while the 30 is for the test.

# Decision Tree

A well-known classification technique called decision trees divides the feature space into regions, each of which is assigned a class label. Recursively partitioning the data using Decision Trees until all regions are pure.

I had to transform some of the data so I could improve the accuracy of my decision tree.

For example, encoding the categorical data was hard. This is because, Categorical variables cannot be handled directly by decision trees. Categorical variables must therefore be encoded as number values. Categorical variables are typically encoded using one-hot encoding, label encoding, and binary encoding. So, to do this, I had to use the get\_dummies() function so I can use the one hot encoding method.

When I originally was training the data on the basic decision tree, using the test data to predict the accuracy, I got an accuracy result of 73.5%. I felt like this was a bit too low, so to increase the accuracy, I set the new parameters to have a :

max\_depth = 5. - min\_samples\_split = 13. - min\_samples\_leaf = 4.

These parameters could better represent my data and get a higher accuracy result. Using these parameters on the data again. The accuracy actually rose to 78%, which is a 4.5% increase without the parameters.

After that, I comducted hyperparameter tuning to get the best parameters, so we can increase the accuracy. I used a grid seatch object so it can scan every possible parameter and give me the best ones for the best results. The parameters they gave me where:

clf\_32: classifier is tuned,

param\_grid,

scoring,

cv

With these parameters, we managed to get an accuracy level of 80% which is a 1% increase with the parameters.

# k-Nearest Neighbour

The K-nearest Neighbour model (k-NN) is a kind of supervised learning technique, which will be applied to my classification task. It is a non-parametric algorithm, which means that it makes no assumptions about the data's distribution. So instead of it being based on the data, it bases its predictions on how much the training data and the input data resemble each other. The distance between each input data point and each training data point is calculated by the k-NN algorithm. Several distance metrics, including the Euclidean distance,(which I used in my model) Manhattan distance, and cosine similarity, can be used to determine the distance.

In the k-NN, I had to use normalisation, by scaling my model correctly otherwise I would get wrong predictions. The scale of the input variables affects how well the k-NN algorithm performs. As a result, scaling the numerical variables can help the k-NN algorithm perform better. To scale this, the. Because StandardScaler is severely disrupted by outliers and because some features have a range of values, MinMaxScaler is applied to the data.

Originally, training the data on the basic decision tree, using the test data to predict the accuracy, I got an accuracy result of only just 71.6%. There are techniques for hyperparameter tuning that can be used to improve this precision, one of these being the grid search cross validation method. This is used to find the best parameters to search over. Within the code, I used the best\_estimator\_’ attribute of the ‘GridSearchCV’ object so that when we are doing the hyperparameter search, the object will return the best performing model, and the get.Params() method prints it out.

After loading and going through all parameters, the method gives us the best parameters we can use to get more accurate results. The ones they printed out was:

'algorithm': 'auto',

'leaf\_size': 30,

'metric': 'minkowski',

'metric\_params': None,

'n\_jobs': None,

'n\_neighbors': 30,

'p': 2,

'weights': 'uniform'

After using the suggested parameters, getting the accuracy again, I managed to get the accuracy to 79% which is huge compared to the previous 71%, gaining an 8% increase after using hyperparameter tuning.

# Logic Regression

The final classification model that I used is the logic regression. For analysing data that yields a binary result, such as determining whether a person has a low or high income, logistic regression is frequently used. As a result, it is regarded as a suitable classification approach for the activity being carried out.

Once again, normalisation is necessary with logistic regression as without scaling of features the model may produce wrong predictions. Since the scale of the input variables affects how well the logistic regression algorithm performs. As a result, To prevent this, scaling the numerical variables can help the Logistic regression algorithm perform better. To scale this, the. Because StandardScaler is severely disrupted by outliers and because some features have a range of values, MinMaxScaler is applied to the data.

In logic regression, I managed to produce the predicted labels and then compare them with the true labels, using the accuracy score. This offers a quick way to gauge how well the model works with fresh, untested data. By doing this, I got an accuracy score of 81.4%. once again, we did hyper parameter tuning, to get the best parameters we could use. These were:

‘C’: 1,

‘Penalty’: ‘12’

‘solver’: ‘libliner’

The Logic Regression took 33 minutes to load, but when it did and when these parameters are used and applied to the test data once more, the accuracy falls by 2%, or by 79%. As logistic regression does not greatly benefit from hyperparameter tuning, this is to be expected.

# Ensemble Learning

Finally, there is one more machine learning technique. A machine learning technique called ensemble learning for classification combines different classifiers to increase the precision and robustness of the classification model. Ensemble learning is based on the notion that by combining the predictions of various classifiers, we can lessen the effect of individual classifier errors and enhance the model's overall performance.

Ensemble learning has multiple methods like bagging, Random Forest, Boosting and stacking but the one that I am going to use is bagging. What is bagging?

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these weak models are then trained independently, and depending on the type of task—regression or classification, for example—the average or majority of those predictions yield a more accurate estimate.  (What is bagging | IBM)

Decision Tree Bagging

To increase the accuracy even further, the bagging classifier is altered with various parameters. This is that the random\_state is set to 42 as this causes the result to stay the same.

The bagging classifier is then equipped with these parameters and trained using the data. The accuracy predicted using the test data was 78%. This indicates that, in comparison to the best decision tree classifier model, bagging had exactly the same accuracy.

Logic Regression Bagging

Once again, to increase the accuracy even further, The bagging classifier is altered with various parameters. This is that the random\_state is set to 42 as this causes the result to stay the same.

.The classifier was trained using normalised test data and MinMaxScaler normalised training data. The outcome of applying these parameters to the classifier for bags is not significantly better than the best logistic regression model without bags. The bagging classifier is then equipped with these parameters and trained using the data. The accuracy predicted using the test data was 81.3%. This indicates that, in comparison to the best decision tree classifier model, bagging had an increase of 2.2% accuracy.

k-NN Bagging

Finally, to increase the accuracy of this bagging even further, The bagging classifier is altered with various parameters. This is that the n\_neighbours was equal to 5.

The bagging classifier is then equipped with these parameters and trained using the data. The accuracy predicted using the test data was 79.02%. This indicates that, in comparison to the best decision tree classifier model, bagging had an increase of 0.02% accuracy.

The final model that I used is the voting classifier. The Voting Classifier uses a majority vote system to combine the predictions from the three classifiers. To specifically state that the final prediction is dependent on the most prevalent class predicted by the base classifiers, the "voting" parameters is set to "hard." Given that it combines all three theories, this is the outcome that was anticipated. The outcome is comparable to the three models' 90.6% mean.

# Section 5 – Evaluation of Machine Learning Models

After executing all of these learning models, to compare them, we used machine learning models, and saw various evaluation techniques can be used. The main one being accuracy in tuning the model. In this section, we’ll also look at confusion matrices show the model's outputs as they are delivered. and a ROC curve to finish off.

## Accuracy

Throughout this coursework, we have been trying to see the proportion of correctly predicted outcomes to all predictions. Since accuracy provides a numerical percentage of how well the models predicted, it is useful when models are improved through tuning. Since accuracy is expressed as a single percentage, it may be quickly assessed when contrasting different models.

Firstly, looking at the classification accuracies s and comparing them with each other, the one that obtained the highest accuracy was in fact the logic regressions model, being 81% The closest to that accuracy was 79%, which is what most of the bagging models received. This demonstrates that, overall, compared to the other classification and ensemble models, the logic regression models were more accurate. However, with hyperparameter tuning, the accuracy falls by 2%, or by 79%. As logistic regression does not greatly benefit from hyperparameter tuning, this is to be expected. This is why logic regression is the best model we used.

The tuned decision tree model has the lowest classification accuracy at 78%. But of all the models, this one produced an accuracy in the shortest amount of time. The logic regression model took 30 minutes to complete, whereas this took only seconds. Decision tree models also have several other restrictions that might result in low accuracy, such as the restriction to questions having binary outputs. Because I set the max depth parameter so low, it had a significant impact on the number of questions the model could answer about the data. This meant that the max depth parameter might have a significant impact on the outcomes. The decision tree produces an accuracy of 78% when the bagging classifier is applied to it, which is the same as before. This demonstrates even more how inferior this classification is to the others.

Next is the k-NN classification model Originally, training the data on the basic decision tree, using the test data to predict the accuracy, I got an accuracy result of only just 71.6% but after using the suggested parameters, getting the accuracy again, I managed to get the accuracy to 79% which is huge compared to the previous 71%, gaining an 8% increase after using hyperparameter tuning. This shows that hyperparameter tuning had the biggest effect on the k-NN model. This model is around about in the middle compared to the others.

Finally, voting ensemble mode the outcome is comparable to the three models' 90.6% mean. This was by far the highest and best accuracy classification we had however, since this model has been made, it cannot be a better model to utilise, but it can yield better results with other evaluation measures.

## Confusion Matrices

Confusion matrices are a tool for visualising a machine learning model's performance. These are matrices that show how many accurate positive, inaccurate positive, accurate negative, and accurate negative predictions the model has made. The confusion matrices for the classification under development will only consist of a 2x2 table that displays the proportion of times the developer's income bracket was properly predicted by the model. Moreover, colour mapping helps improve the tables' data visualisation.

## Decision Tree Classifier

Chart, treemap chart

Description automatically generatedThe decision tree model predicted correctly more often than not when the tuned decision tree classifier was applied to a confusion matrix. The fact that the tiles in the "main diagonal" have higher values demonstrates this. In addition, the model generated 386 more false-negative findings than false-positive results (168 results). This demonstrates that the model favours the condition being predicted as being twice as likely to be negative as it would be to be positive.

## k-nn Classifier

The k-NN model predicted correctly more often than not when the tuned k-NN classifier was applied to a confusion matrix. The values on the high income diagonal are very similar, which shows there was an equal number of predictions of negative and positive In addition, the model generated 1456 more True-negative findings than false-positive results ( 569 results). This demonstrates that the model favours the condition being predicted as being twice as likely to be negative as it would be to be positive.

Chart, treemap chart

Description automatically generated

## Logic Regression Classifier

The Logic Regression model predicted correctly often when the tuned Logic regression classifier was applied to a confusion matrix. The values on the false diagonal are very similar, which shows there was an equal number of predictions of negative and positive In addition, the model generated 1256 more True-negative findings than false-positive results ( 457 results). This demonstrates that the model favours the condition being predicted as being twice as likely to be negative as it would be to be positive. Graphical user interface, application

Description automatically generated

## ROC Curve

A graph that illustrates the performance of classification models using the two parameters true positive rate and false positive rate is called a receiver operating characteristic curve, or ROC curve. The confusion matrices' values can be used to derive these two numbers. This pair then be plotted onto a line graph to get the AUC.

Chart, line chart

Description automatically generated

Looking at the ROC graph and the AUC values, Logic regression is the highest with an AUC of 0.85, with the decision tree having 0.79 and the k-NN only having 0.70.

# Section 6 – Conclusion on Machine Learning Coursework

## Results

After all the analysis I believe that after examining many performance measures, it can be said that the Logic regression performed the best in terms of precision and accuracy. The bagging approach and hyperparameter tuning might be used to further improve this classifier, which raised its accuracy and precision. The factors had a significant impact on this model and significantly enhanced it. The tuned decision tree was the model that performed the poorest, but the k-NN wasn't much better because it produced poorer outcomes than the decision tree, as seen by the initial accuracy and the ROC graph. Yet, the decision tree was still generally the worst.

This coursework was useful, as I could compare the data and use many different methods to help me with finding the income level of developers, there was many useful models.

I now understand Crisp-DM a lot more than before and have actually gained significant knowledge on how to handle ,y data and conduct an EDA. Before reading and understanding this work, id have no clue in what to do but due to this coursework, I feel very confident. Now I just hope to extend my knowledge and improve for the future.

## References

*What is bagging?* (no date) *IBM*. Available at: https://www.ibm.com/topics/bagging (Accessed: March 29, 2023).

Real Python (2021) *Visualizing data in python using Plt.scatter()*, *Real Python*. Real Python. Available at: https://realpython.com/visualizing-python-plt-scatter/ (Accessed: March 29, 2023).

InfoPython (2022) *How to plot a pie chart using Python (9 examples)*, *InfoPython*. Available at: https://infopython.com/how-to-plot-a-pie-chart-using-python/ (Accessed: March 29, 2023).

Nik (2022) *Pandas crosstab - The Complete Guide (w/ examples) • datagy*, *datagy*. Available at: https://datagy.io/pandas-crosstab/ (Accessed: March 29, 2023).

Carvalho, T. (2022) *Visualizing clusters with Python's matplotlib*, *Medium*. Towards Data Science. Available at: https://towardsdatascience.com/visualizing-clusters-with-pythons-matplolib-35ae03d87489 (Accessed: March 29, 2023).

Brownlee, J. (2021) *How to develop a bagging ensemble with python*, *MachineLearningMastery.com*. Available at: https://machinelearningmastery.com/bagging-ensemble-with-python/ (Accessed: March 29, 2023).

*Optimization of hyper parameters for logistic regression in python -* (no date) *ProjectPro*. Available at: https://www.projectpro.io/recipes/optimize-hyper-parameters-of-logistic-regression-model-using-grid-search-in-python (Accessed: March 29, 2023).

## Appendices

Decision Tree

Chart

Description automatically generated