# Artificial learning & Machine learning

# Coursework 1

# N0992216

## Contents

Introduction …………………………………………………………………………………………………………………….3

Data Understanding ………………………………………………………………………………………………………….4

Data Cleaning ……………………………………………………………………………………………………………………5

Exploratory Data Analysis ………………………………………………………………………………………………….7

Cluster Analysis ………………………………………………………………………………………………………………..15

Machine Learning for classification …………………………………………………………………………………..19

Evaluation …………………………………………………………………………………………………………………………22

Conclusion ………………………………………………………………………………………………………………………..25

Appendix …………………………………………………………………………………………………………………………..26

Reference ………………………………………………………………………………………………………………………….28

## Introduction

Cross-industry standard process for data mining, CRISP-DM, is an open standard process model used by data miners to work on analysis projects and is one of the most popular. CRISP-DM methodology is described in terms of a hierarchical process model, comprising four levels of abstraction phases, generic tasks, specialized tasks, and process instances (Wirth, R. and Hipp, J., 2000). Level one, phases consist of multiple generic tasks. Level two, generic are tasks designed to be complete and stable as described by Wirth and Hipp (2000). Complete meaning covering the process of data mining and stable means the model is valid for any unexpected circumstances. Level three, specialise, describes how a generic task should behave in a specific situation. Finally, level four, process instance, is a record of results from a data mining engagement. These are taken from the higher levels.

1. **Business Understanding**

Any good project starts with a deep understanding of the customer’s needs (Hotz N, 2023). This stage focuses on understanding the objectives and requirements of the project as a business. Project goals are set, assessing the situation for factors that could affect the project and producing a project plan is key steps in this stage.

1. **Data Understanding**

This stage involves collecting data, using exploratory data analysis to analyse the data, and developing initial insights. Finally, evaluate the quality of data and select interesting subsets that may contain patterns.

1. **Data Preparation**

Hotz (2023) estimates 80% of the project should be data preparation. This phase prepares the data set for modelling. It involves selecting, cleaning, constructing, integrating and formatting the data.

1. **Modelling**

For this stage, a data model is selected to use on the data set. Often several modelling techniques can be used on the same data. we must calibrate the model settings for optimal results. This step may result in us having to loop back to the data preparation stage to adjust the data for optimal results.

1. **Evaluation**

This stage is especially important for the future of the project. Evaluate if the model meets the business objectives. Review the process and see if anything was missed. Summarise findings and ask if anything was missed. Finally, determine the future of the project.

1. **Deployment**

In this stage, a final report is collected and made. The process is reflected on and reviewed.

Knowledge gained will have to be presented in a way customers can use (Chapman et al, 2000).

In this coursework, I intend to analyse the ‘2021 New Coder Survey’ data set collected by freeCodeCamp. FreeCodeCamp is a non-profit organisation that helps make learning how to code accessible to everyone. I will apply exploratory analyse on this dataset, of 18000 people, using python programming. By analysing this dataset, I plan to better understand this data. The focus of the project is to predict whether the developer is high-income or low-income based on the survey data provided. By the end of the project, I intend to build my confidence in handling and analysing real-world data, as well as gain a better understanding of the CRISP-DM methodology.

It is important for my project to produce accurate results. Hence, I will be using the CRISP-DM methodology as it is the most appropriate methodology for this project. Furthermore, it will assist in structuring the data before applying any models to it.

**Project Plan**

I will start by going through the survey and questions and understanding them. Secondly, I would pick my predictor columns and the target column which is the income column. Next, I would prepare the data by cleaning it. This is done by removing null values, outliers, and incorrect data. As I would like to retain as many columns as possible I am going to try to replace these values with the mode, median and mean. After to further understanding the columns and trends, I am going to do exploratory data analysis. This will involve creating one, two, and three-variable graphs to visualise the data. I will then move to data clustering to find groups within the data. Finally, I will do the classification, which predicts and use multiple models. I will then print the accuracy to find the best model.

**Success Criteria:**

* Attributes used for the dataset are understood and explained clearly.
* Data fully cleaned.
* EDA identifies patterns and relationships between attributes.
* Clustering used to describe characteristics between High and low income.
* Use multiple models for prediction.
* Evaluation of all models and pick the most accurate model.
* A conclusion written to summarise all findings.

## Data Understanding

The data I am analysing in this project was collected by freeCodeCamp. This survey was published in December 2021 and contained more than 18000 responses. The survey was made open so there are a variety of responses from many different people. The survey asked a mixture of open and closed questions. All to find out how and why each chose to learn how to code.

Their many questions but I have seen 4 prominent categories. These include location, resources, expectations, and background. Location questions reveal where new coders are coming from, resource questions cover what resources they are using to learn, expectation questions ask what presumptions they have on working patterns, pay etc, and background questions show what careers they have had before learning to code. Some questions have missing or null values, but appropriate measures will be used to combat this. The survey gives me a large data set to analyse, which will enable me to get reliable results. Hence this data set I extremely useful.

**Predictors Selected**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Attribute Name | Description | Data Type |
| 7 | Hours\_Learning | About how many hours do you spend learning each week? | Float |
| 8 | Months\_Programming | About how many months have you been programming? | Float |
| 19 | Relocate | Are you willing to relocate? | Object |
| 23 | Age | How old are you? | Float |
| 25 | Race | With which of these groups do you primarily identify? | Object |
| 26 | Area | Which part of the world do you live in? | Object |
| 32 | Education\_Level | What is the highest degree or level of school you have completed? | Object |

The target variable is question 22 which is ‘About how much money did you earn last year from any job or employment (in US Dollars)?’. A high-income earner is described as an income over $29999 otherwise the individual is classified as a low-income earner.

**Opinions on predictors**

These predictors will predict whether an individual will be a high-Income or a low-Income earner. I chose these predictors specifically because I believe they have a large influence on income. For example, if you are willing to relocate you are more likely to have a high income or if you live in south Asia you are more likely to have a low income etc.

Attributes that are most significant to the target include respondent demographic, education level, and time spent learning. These are the most important attributes. However, relocation information is also very important.

**Target**

The target selected was income. Using this the model will predict if the individual is classified as high-income or low-income. In this case, we are defining a high income as an income over $29,999.

## Data Cleaning

## 

**Figure 1 – shows the head of the data frame.**

Raw data is often dirty. It contains missing values, incorrect values, and outliers. Furthermore, the data formats may not be suitable for machine learning models. In Figure 1, we can see that there are null values, NaN. Hence, data cleaning is needed to remove these values and make them more accurate for the machine learning model to predict. A lot of missing values will interfere with the data analysis.



**Figure 2 - shows the total rows.**

Text

Description automatically generated

**Figure 3 - shows the number of missing values in each column.**

* Hours\_Learning has a response rate of 91.6%
* Months\_Programming has a response rate of 92.1%
* Age has a response rate of 94.4%
* Relocate has a response rate of 95.8%
* Race has a response rate of 94.9%
* Area has a response rate of 95.1%
* Education\_Level has a response rate of 94.9%

For Hours of Learning per week, we had 1512 null values. We could drop all the missing values however this would be wasteful and remove 1512 rows from the table. As a result, I have decided to replace the null values with the mean. Furthermore, the highest number of hours learning per week is 148 however there are only 168 hours in a week. This is an unrealistic answer and I consider it an outlier. As a result, I have decided to make the maximum number of hours learning per week 101. This is because the average sleeping hours for an adult is 7-9 hours. Using the lower bound of 7 means that the maximum number of hours sleeping per week is 49. Hence, I will be replacing all values above 101 hours with the median of the data. I am using the median because the data is skewed due to the outliers.

For the numerical values I will be using the median however for the categorical data, I will be using the mode. This will allow for more accurate data analysis. As a result, when predicting at the end of the project, the predictions will be more accurate.

The most complex attribute that I select is race. Race is a very complex attribute that has 587 unique values. values. Examples of the race column entries are ‘Human’,’ Black/African’,’ Pacific Islander,’ Tamilian (South Indian)’ etc. There are incorrect values, null values and a lot of data which can be grouped. Therefore, I decided the group these values into 6 groups: Asians, Black/African American, Hispanic/Latino, Mixed Race, Other and White. This cleaning will allow for easier exploratory data analysis, clustering, and classification.

Text, letter

Description automatically generated **Figure 4 – A section of the Race’s unique values.**

## Exploratory Data Analysis (EDA)

Exploratory Data Analysis is an approach used to analyse data by visualising the data. This can help identify outliers and discover patterns. Similarly, through EDA we can see the spread of answers and easily analyse it.

**Numerical Data**

**Age – How old are you?**

**Chart, box and whisker chart

Description automatically generated**

**Figure 5 – Age plot.**

To start I will be analysing all my numerical data. Starting with the Age attribute. We can view this through a box plot (see figure 5). After removing the major outliers e.g., 120, We can see the survey has been answered by a range of ages. The youngest is under 10 years old, the highest is 80 years old and the median is around 23. In figure 5, we can see that many ages over 50 years old are seen as outliers. However, I have chosen to accept them because learning to code can begin at any age.

**Hours\_Learning – About how many hours do you spend learning each week?**

**Chart, box and whisker chart

Description automatically generated**

**Figure 6 – Hours Learning per week plot.**

Next, we have the Hours learning per week attribute. The median is around 8 hours per week of learning. The figure above has a lot of outliers, in the cleaning section above I explained how learning over 100 hours a week was unrealistic, as a result, I will be accepting all the outliers. We can see the upper and lower bounds are 15 and 5 hours per week which is realistic especially taking into account the spread of ages. Individuals over 21 are more likely to have commitments and hence have less time to learn.

**Months\_Programming – About how many months have you been programming?**

**Shape

Description automatically generated with low confidence**

**Figure 7 – Months spent programming plot.**

In figure 7, we see the boxplot for months of programming. The median being near 0 shows us that the majority of the people who answered the survey are new to coding. Considering the spread of ages (see figure 5), we can accept all outliers above the top whisker.

**Chart, histogram

Description automatically generated**

**Figure 8 – scatter matrix.**

A graph that is useful for finding trends between numerical data types is a scatter matrix (figure 8). Using a scatter matrix, I can see the correlation between 3 attributes: age, hours learning per week and months of programming. Firstly, I will analyse the age and hours of learning per week. In the figure, we can see as the older the individual the fewer hours they spend learning per week. This shows a negative correlation. Furthermore, if we analyse hours of learning per week and months spent programming, we also see a much weaker negative correlation. The higher the number of months of programming, the fewer hours spent learning per week. Finally, age and the number of months spent programming have no real correlation. Seen in figure 8.

Chart, scatter chart

Description automatically generated

**Figure 9 – Age/Hours learning per week and high-income scatterplot.**

In figure 9, we are analysing 3 variables. One variable is the target high income. The other being hours of learning per week and age. Here we can see the majority of high-income people are between the ages of 20 and 70 and learn less than 20 hours per week. However, apart from this, the data does not have a strong correlation.

Chart, scatter chart

Description automatically generated

**Figure 10 – Age/Months Programming and income scatterplot.**

In figure 10 we compare months spent programming with age. Similar to figure 9, in figure 10, we can see the majority of high-income earners are between the ages of 20 and 70. However, we can see a spread of individuals from 0 months spent programming to over 500 being high-income earners. This can show that the months spent programming attribute has a lesser effect on whether the individual; is a high-income earner or not.

**Categorical Data**

**Relocate – Are you willing to relocate?**

**Chart, pie chart

Description automatically generated**

**Figure 11 – Relocate Pie Chart.**

In figure 11 we can see that over half of the respondents are willing to relocate for a job. This is followed by maybe which is at 35.7%. I decided to keep maybe as a valid response as some respondents may not have a definitive answer to if they want to relocate or not.

A picture containing chart

Description automatically generated

**Figure 12 – Relocate, Age and High-Income Strip plot.**

In figure 12, we compare age, relocation, and high income in a strip plot. This graph has weak correlations between the variables. However, if we look closely at the no strip, we can see that it has slightly more high-income earners at younger ages. This may reveal that individuals that do not relocate have a high chance of becoming a high-income earner earlier.

**Race – With which of these groups do you primarily identify?**

**Chart, pie chart

Description automatically generatedChart, bar chart, histogram

Description automatically generated**

**Figure 13 – Race plot. Figure 14- Race Bar chart.**

In figure 13, we can see the percentage of respondents of each race. The highest is people who identify as white with 40.7% the lowest being respondents who chose other 0.5% of the survey. In figure 14, we can see the actual number of respondents. This will help us with expressing percentages when comparing with other variables. This clearly shows over 7000 white respondents took the survey followed by about 5000 Asian respondents. This shows the variety of races that took the survey.

**Chart, histogram

Description automatically generated**

**Figure 15 – High-Income and Race Bar chart.**

Following, figures 13 and 14, figure 15 shows how many high-income earners are in each race. We can see that out of 7000 white respondents, 2000 of them are high-income earners. This means that 28% of them are high-income earners. Whereas with Asian respondents it is only 7.5% and with black/African American respondents there is only 12.5%. Hispanic/Latino has a slightly higher percentage at 13.3%. However, there is an overwhelming gap between white high-income earners and the other races. This infers that high-income earners are often white.

**Area – Which part of the world do you live in?**

**Chart, pie chart

Description automatically generated**

**Figure 16 – Area pie chart.**

In figure 16, we can see that most people who took the survey live in North America. Followed closely by Europe and Central Asia. This is the trend I expected from the areas in which respondents lived. From the figure above we can see we have quite an even spread of places in the world. This will allow our predictions to be more accurate.

**Chart

Description automatically generated**

**Figure 17 – High-Income and Area bar chart.**

In figure 17, we can see how the area in which people live compares to how high-income. As suspected the high-income earners are mainly situated in North America. The next highest number of high-income earners is in Europe and Central Asia. Then east Asia and the pacific which is surprising as they had the lowest percentage of respondents living there, at 2.9%.

**Education\_Level – What is the highest degree or level of school you have attended?**

**Chart, pie chart

Description automatically generated**

**Figure 18 – Education level Pie chart.**

In figure 18 above we can see the highest level of education each respondent achieved the highest being a bachelor’s degree at 35% which was to be expected. However, the next highest is a high school diploma at 32.4 %. There is only a difference of 2.6% between these two. There is a spread between the rest of the levels, the lowest being people who have achieved a PhD. However, the vast majority of the respondents have a degree 49.5%.

## Chart, histogram Description automatically generated

**Figure 19 – Age, Months programming and Education Level stacked Bar chart.**

In figure 19 we can see that respondents who study bachelor’s degrees spend the most time programming. This is expected, however, the next most are High school diplomas which is surprising. After 25 years, the higher the respondent’s age the lower months spent programming. This is surprising because although the tech industry is developing, it is not new. I expected the months spent programming to be highest between the 20-33 range. However, the highest spike is 24 years. Altogether, we can see the most months spent programming is at 24 years.

## Chart Description automatically generated

**Figure 20 – High-Income and Education Level Bar chart.**

In figure 20, we can see that most high-income earners have a bachelor’s degree and the lowest have not finished high school. This was the expected trend. Respondents with degrees have the most high-income earners.

## Cluster Analysis

Clustering is a data mining process for discovering groups and identifying interesting new patterns in underlying data (Frades, I. and Matthiesen, R., 2010). Cluster analysis allows us to better understand the respondents by identifying individuals with similar traits to one another.

The data will be split into two subsets based on income. These subsets will determine whether an individual is a high-income earner or a low-income earner. A high income is defined as over $29,999 or income level 10 and above. A low income is defined as under income level 10. We will then create a data frame with the target and some, not all, of the attributes. The categorical data will have to be encoded using the one hot ender. This will turn the categorical data into numerical data. Cluster analysis techniques will then be performed on the attributes to spot trends within the data.

**K-Means**

K-means clustering is a powerful machine-learning technique. Kodinariya, T.M. and Makwana, P.R. (2013) describe it as a simple and fast clustering technique that provides the optimal number of clusters in advance.

The simplest K-means clustering goes through these steps. Firstly, asks the user how many clusters the data should be partitioned to, in this case, k. Then, by random assign k records to the initial cluster centre location. Then, for each record, the algorithm finds the nearest cluster centre, generating k clusters. For each of the k clusters, we find the cluster centroid and update the location of each cluster centre to the cluster centroid. The algorithm repeats this until convergence or failure.

K-Means has a number of parameters that can affect the result of the analysis. The ones used in my program:

* n\_clusters - This sets the number of clusters. N\_clusters is set to 3 in the first subset and 4 in the others. This makes the clusters easier to read and easier to compare to one another. Furthermore, in figures 21,22 and 23 the elbow methods show that the optimal number of cluster is 3,4 and 4 respectively.
* random\_state - This determines random number generation for centroid initialisation. It is useful when we want to create the same clusters every time.

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

**Figure 21 – Elbow for first cluster subset. Figure 22 – Elbow for second cluster subset.**

Chart, line chart

Description automatically generated

**Figure 23 – Elbow for third cluster subset.**

Firstly, I decided to do a cluster analysis on the area, race, and income level attributes.

Chart

Description automatically generated

**Figure 24 – Silhouette Plot for the area, race and income level clustering.**

To further make sure the number of clusters selected was appropriate, I used the silhouette method. This is because the elbow method only considers cohesion. Whereas the silhouette method takes into account both cohesion and separation. With the silhouette method, we are looking for a silhouette score above 0.5, as this is considered strong. By doing this I have concluded that 3clustersr is a good amount for the analysis, as it has the highest score of 0.614.

Chart, bar chart

Description automatically generated

**Figure 25 – bar plot cluster analysis on the area subset.**

From figure 21, we can see in all clusters North America is the majority. Both clusters 0 and 2 have similar patterns, showing North America and Europe and Central Asia have the highest counts. However, Cluster 1 is different. It shows south Asia is the second highest count. Cluster 1 has more diversity in living areas than the other clusters. Cluster 1 is the least diverse, most of the count is in North America.

Chart, bar chart

Description automatically generated

**Figure 26 – bar plot cluster analysis on the race subset.**

From figure 22, we can see in both clusters 0 and 2, the most dominant race is white. Looking back at figure 21, this makes sense and the majority of these clusters were from North America and Europe. Cluster 0 is less diverse than cluster 2 but there is not a major difference between the two. However, in cluster 1 the highest race is Asian, followed closely by white. This cluster is much more diverse in race, this is likely because it is the most diverse in the area as well.

Chart, histogram

Description automatically generated

**Figure 27 – bar plot cluster analysis on the income level subset.**

In figure 23, we can see there is a spread between income levels among all three clusters. Cluster 1 has the lowest income ranging from 0.0 to 4.0. This cluster was the most diverse in area and race. Since their income level is under 10 they are classified as low income. Cluster 2, has a higher income than cluster 1 but is still not classified as high income. Their income ranges from 5.0 to 9.0. Finally, Cluster 0 has the most income and is classified as high income. This cluster is also the least diverse in race and area lived. This can imply that area lived in and race has an impact on income level. The highest earners are white and in North America.

The next cluster analysis I wanted to do was education level and income level.

**Chart

Description automatically generated**

**Figure 28 – Silhouette Plot for the education level and income level clustering.**

In figure 28, we can see the silhouette method for the below cluster analysis. The silhouette score is 0.627 showing 4 clusters is a good amount for the analysis.

Chart, bar chart

Description automatically generated

**Figure 29 – bar plot cluster analysis on the education level subset.**

In figure 29, we can see a similar trend between all clusters. The highest of each cluster is a bachelor’s degree and a high school diploma. Through this, it will be difficult to find a relationship between income level and education level. This is because all clusters have a similar starting pattern, the only difference is the count in each cluster. If we compare appendix D and figure 29 it will be difficult to find a direct correlation.

The final cluster analysis I did was looking at the relationship between if an individual is willing to relocate and income level. However, as shown in appendix A, the four clusters expressed the same pattern and different counts. Using appendix B, this will be difficult to find a direct correlation between the two.

## Machine Learning for Classification

**ML Workflow**

Figure 30 is a flowchart which visualises the phases necessary to get accurate models for the investigation. The process starts with collecting the data and preparing it, which includes cleaning. It is then split into training, testing and validation data. Some models only need training and testing, however, we will be using validation data to find the most accurate model. We create the model using the training data and then test it using the testing data. If the accuracy could be improved then tuning will be applied using the validation data. Finally, the model is evaluated and deployed for prediction.

Diagram

Description automatically generated

yes

no

**Figure 30 – flowchart showing the workflow of machine learning for classification.**

The models I chose were linear regression, K-nearest neighbour, decision tree and ensemble learning Voting Classifier. To tune, I used the holdout method, bagging and grid search on some of the methods.

**K-Nearest Neighbour**

K-nearest neighbour is a supervised learning classifier. With KNN, firstly, we have a set of instances with known labels. Afterwards, for the new samples, we choose k instances that are very similar to the new sample. Lastly, assign the class label to the new sample based on its neighbour instances label. If we choose a small k-value it may lead to overfitting the data. This will mean the model will not be able to achieve generalisation, it will be trained mostly on one dataset. If the k-value is too large individual values will be overlooked. Some values may be much bigger than others hence, I will be normalising the data. In my case, I used the MinMax Scalar and fitted it to both the X train and X test values.

KNN cannot use categorical values for the model. As a result, I will have to encode the categorical data I have using the one hot encoder. The parameters I am using in the KNN model are weight and metrics. I will be setting the weight to “uniform” this ensures all points in each neighbourhood are weighted equally. I will be setting metrics to “Euclidean” as it is a measurement to find the shortest distance between two consecutive points.

I will be using KNN with the hold-out method to tune the hyperparameters. Doing this may significantly improve the performance of the KNN model. Hold-out method is a manual tuning method. Firstly, I will divide the data into testing, validation, and test data. In Sklearn I cannot split the data directly into 3 parts, instead, I will split it into a training and testing set followed by a training and validation set. I will fit the model on the training data and predict the accuracy of the validation data. Finally, I will record the accuracy in an array and plot the graph to see if it truly improves the accuracy.

I will also use the grid search method to tune the hyperparameters. Grid search is a more powerful automatic tuning technique. This should produce better results than the hold-out method. Grid search will do an exhaustive search over the parameter values for an estimator. This will be specified with the param\_grid parameter. Once the search is complete, it outputs the hyperparameters which produced the most accurate model.

Lastly, I used bagging with the KNN model. The bagging classifier can be on KNN as it has a high accuracy score. We can make this score even higher by changing parameters. By setting n\_estimators to a higher number, in my case 11, it means the model is run 11 times. Secondly, random\_State is set to 1 which causes the result to stay the same. These parameters are put into the bagging classifier and trained on the data. The test data predicts the accuracy.

**Decision Tree**

A decision tree is a collection of decision nodes. The decision nodes are connected by branches extending downward from the root node. It starts with the root node, the attributes are tested at decision nodes and each possible result is a branch. Each branch leads to a decision or leaf node. Training data must be labelled since it is a supervised learning method, so the target high-income must be labelled. We do not need to apply normalisation to the data when creating a decision tree. This is because it does not interfere with the prediction made.

I will be using two parameters for the decision tree model. The criterion measures the quality of a split and will take in the value ‘gini’. this will measure the purity of a feature and take the purest feature for splitting. The other parameter I will be using is Max\_depth which will take the integer ‘3’. This holds the maximum depth of the tree. Decision trees cannot use categorical data, so I will be using the one hot encoder to transform the data into numerical data.

Like above in the KNN model, I will be using the hold-out method to manually tune the model. Again, by splitting the data into 3 testing, validation, and training data. I will continue to use criterion = ‘gini’ but, the max\_depth will be a range from 1 to 16. Then I will record the accuracy and see if it is better than the original model.

Then I will also use a grid search to do automatic tuning of hyperparameters. All the possible hyperparameters will be identified and the grid search will combine all the different combinations. Once the search is complete the combo with the most accurate result will be produced and the accuracy will be outputted.

**Logistic Regression**

Logistic regression is a logistical machine learning model. It is best for data that has binary outputs as it uses the sigmoid function which has a range between 0 and 1. For my high-income target, it is perfect. However, the downside of logistic regression is that it assumes there is no error in the output attribute, hence, removing erroneous data is needed. Similarly to KNN and decision trees, logistic regression cannot use categorical data. So I will be using the one-hot encoder to change this to numerical data. Furthermore, normalisation is needed for the model. Without this, the model may make incorrect predictions. I shall be using the MinMax scalar to normalise the data. The standard scalar is not a good decision because it is heavily influenced by outliers.

Furthermore, I will be using the grid search to tune hyperparameters like KNN and decision trees. The only parameter in the gird is the penalty which equates to l2. The parameters for the logistic model will be solver and max\_iter. Solver will be set to ‘Ibfgs’. This is because it supports the l2 and it is good for multiclass problems. Max\_iter is simply the number of iterations taken to for the solver to converge. This will be set to 1000. All the combos will then be found and the accuracy of the most accurate will be outputted.

**Ensemble Learning**

For the ensemble learning model, I will be using the Voting Classifier model. This will involve combing the predictions of several different base estimators which are built independently. Firstly, the data will be partitioned into test, training, and validation sets. Following this, we will train and validate a set of base classifiers on the training and validation data. Next, we apply the base classifier to the prediction on test data. Finally, we will combine the classification prediction into voting ensemble models, using the voting method. In this case, we are trying to achieve majority classification. This is where the classification has more than 50% of the votes.

I will be using the three previous models for the classifier estimators. This is the K-nearest neighbour, Logistic regression, and decision trees. I decided to normalise the data as bot KNN and logistic regression need it. The program then combines the three base classifiers and averages their results. This should produce high accuracy.

## Evaluation

|  |  |
| --- | --- |
| Model | Accuracy |
| K-Nearest Neighbour | 96.73% |
| K-Nearest Neighbour with holdout method | 96.10% |
| K-Nearest Neighbour with grid search | 98.07% |
| K-Nearest Neighbour with bagging | 98% |
| Decision Tree | 100% |
| Decision Tree with holdout method | 100% |
| Decision Tree with grid search | 100% |
| Logistic Regression | 99.87% |
| Logistic Regression with grid search | 99.76% |
| Voting Classifier | 99.9% |

**Figure 31 – Models used and Accuracies.**

Figure 31 shows the accuracy of each model in percentage. We can see the accuracies range from 96% to 100%, which may reveal we have some successful predictions. To further, back up my claims I will be using the accuracies above and confusion matrixes. A confusion matrix is a table with two rows and two columns. It reports the number of true positives, true negatives, false positives, and false negatives. This can reveal whether a prediction is positive, or a prediction is negative.

Chart

Description automatically generated Chart

Description automatically generated

**Figure 32 – KNN confusion matrix. Figure 33 – KNN holdout method confusion matrix.**

**Chart, calendar

Description automatically generated**Chart, calendar

Description automatically generated

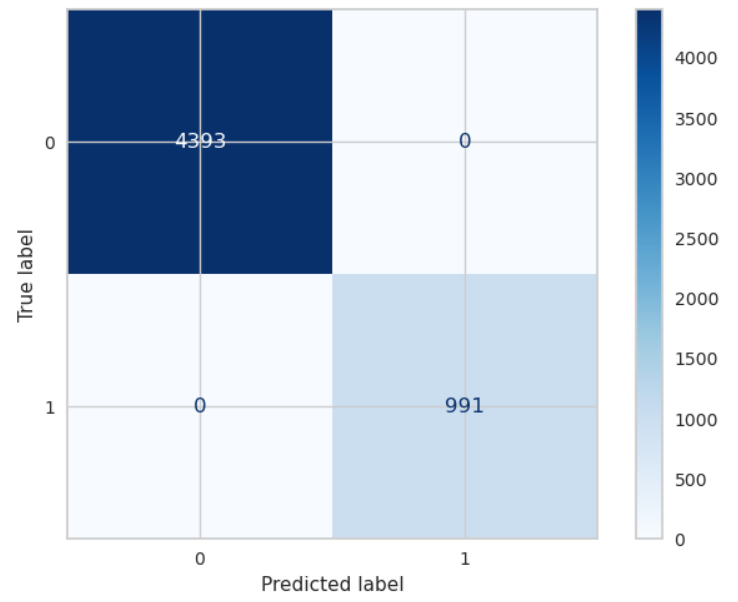
**Figure 34 – KNN with grid search. Figure 35 – KNN with bagging.**

Starting with the K-Nearest neighbour method, the accuracy score is 96.73%. Figure 32, the confusion matrix shows the presence of positive is 4317, which predicts positive. The presence of negative is 76, which predicts negative. This means the number of true positives and negatives is much higher than false negatives. When we apply the hold-out method, the accuracy drops to 96.10%. Making the model less efficient. In figure 33, we can see the presence of positives increases in both true positive and true negative. In figure 36, we can see the training data is overfitting because of the holdout method. I then decided to use the grid search on the KNN model. This made the accuracy jump to 98.07%. In figure 34, we can see the true positives stayed consistent, however, the number of false positives increased for the grid search. Tuning with grid search overall made the KNN model more accurate. Finally, I use the bagging method which leads to a 0.07% decrease in accuracy compared with the grid search. The confusion matrix is the same as the grid search.

Chart, line chart

Description automatically generated

**Figure 36 – line plot with KNN Holdout.**



**Figure 37 – confusion matrix for decision tree.**

For all the decision tree models decision tree, decision tree with gird search and decision tree with the hold-out method. we can see 100% accuracy. In the figure above, 37, we can see there are no false positives or false negatives. In figure 38, we can see the line plot of the decision tree. This is inaccurate. It is impossible for a model to be 100% correct. As a result, this is not a reliable model to use. This is most likely due to overfitting the data.

## Chart, line chart Description automatically generated

**Figure 38 – line plot with decision tree.**

I then investigated the logistic regression. With the basic model, the accuracy result is 99.87%. We can see in figure 39 that there are a few false positives and false negatives. This shows that the model is unreliable. Furthermore, when I used the grid method on the logistic regression, nothing really changed. The confusion matrix stayed the same. The accuracy dropped to 99.76%.

Chart, calendar

Description automatically generated

**Figure 39 – logistic regression confusion matrix.**

Finally, we create an ensemble method, which is the voting classifier. This used the decision tree, logistic regression, and KNN. The accuracy is 99.9% for the ensemble method. This is an inaccurate model. I believe it is because it uses the decision tree which is an inaccurate method. It is impossible for a model to be 99.9% accurate.

The most accurate model that I have created is KNN with bagging with an accuracy of 98%. This is because everything over 98% is considered inaccurate. This is majorly due to overfitting. Values over 98% are not seen as realistic.

## Conclusion

Through looking at the performance of the models, it can be concluded the K-nearest Neighbour model with bagging is the most accurate. Closely followed by K-nearest Neighbour. The worst performance was the decision tree. The decision tree outputted a lot of unrealistic results. Logistic Regression and ensemble learning were better than decision trees but still majorly unrealistic.

From the coursework, I was able to prepare and clean data, perform exploratory data analysis, perform cluster analysis and perform prediction. I feel like I’ve gained a better understanding of CRISP-DM. I found clear trends between some attributes and the target. Before the coursework, I wouldn’t have been as confident. The next step would be to further delve into different cleaning techniques, models and exploratory data analysis graphs.

## Appendix

Chart, bar chart

Description automatically generated

**Appendix – A - bar plot cluster analysis on the relocate subset.**

**Chart

Description automatically generated**

**Appendix – B - bar plot cluster analysis on the Income level subset based on the relocate.**

**Chart

Description automatically generated**

**Appendix – C – Silhouette plot for the relocate and income cluster analysis**

Chart

Description automatically generated

**Appendix – D - bar plot cluster analysis on the Income level subset based on the education level**

**A picture containing timeline

Description automatically generated**

**Appendix D – tree plot for decision tree holdout method**

## References

Chapman, P. et al, 2000. CRISP-DM 1.0 - Step-by-step data mining guide. Available at: http://www.crisp-dm.org/CRISPWP-0800.pdf (Accessed: February 11,2023).

Hotz, N. (2023) *What is CRISP-DM?*, *Data Science Process Alliance*. Available at: https://www.datascience-pm.com/crisp-dm-2/ (Accessed: February 11, 2023).

Wirth, R. and Hipp, J., 2000, April. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining* (Vol. 1, pp. 29-39).

Frades, I. and Matthiesen, R., 2010. Overview on techniques in cluster analysis. *Bioinformatics methods in clinical research*, pp.81-107.

Kodinariya, T.M. and Makwana, P.R., 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal*, *1*(6), pp.90-95.