# Introduction

CRISP-DM methodology (Wirth and Hipp, 20000) which stands for cross-industry process for data mining is a well-proven and robust methodology that provides a well-defined style to planning a data mining project. The method is split into six stages, the first stage is to understand what is to be accomplished from a business perspective, the main goal of this stage is to uncover important factors that could influence the outcome of the project. This can be done by setting primary objectives from a business view, producing a project plan this should specify steps to be performed during the rest of the project. Also, layout the business success criteria that you will use to determine whether the project has been successful from the business perspective.

Diagram

Description automatically generated

Figure 1.1 – CRISP-DM process

The CRISP-DM process outlines the steps involved in performing data science activities from business need to deployment, and most importantly it indicates how iterative this process is and how there is always room for improvement.

CRISP-DM can be applied in the financial industry to detect money laundering for example as this involves sifting through large amounts of data without a specific modeling goal. Instead of modeling it focuses on data exploration and visualization to uncover suspicious patterns in financial data, CRISP-DM would allow you to create a data mining model that fits this need. For such an application in this industry the modeling, evaluation and deployment phases might be less relevant than the data understanding and preparation phases. However, it is still important to consider some of the questions raised during these later phases for long-term planning of detecting money laundering and future data mining goals. In the engineering field CRISP-DM methodologies provides a communication and planning foundation for data analytics within the production domain using IBM extension (ASUM-DM) analytics solutions unified method for data mining analytics.

The data analytic task requires building machine learning models for different scenarios regarding the censum income dataset. CRISP-DM can be applied to the data analytic task by using the six different stages of the methodology in planning the task.

I intend to gain a very in-depth knowledge into the CRISP-DM methodology and each of the different planning stages and how they will apply to the data analytic task.

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

The Adult data set (Censum income dataset) is downloaded from UCI Machine Learning repository (1996) and was extracted by Barry Becker from the 1994 census bureau database found at <http://www.census.gov/ftp/pub/DES/www/welcome.html> the dataset is listing participants and multiple attributes per each participant*(full list of attributes can be found at Appendix A)*. A set of reasonably clean records was extracted using the following conditions:

((AAGE>16) && (AGI>100) && (AFNLWGT>1) && (HRSWK>0)).

The dataset task is to predict whether income exceeds $50K/yr based on census data. The dataset:

Characteristics- Multivariate

Number of Instances (mix of continuous and discrete)- 48842

Duplicate or conflicting instances- 6

Number of Attributes- 14

Attribute Characteristics- Categorical, Integer

Area- Social

Missing Values- YES

The dataset has a large number of instances from the census at 48,842 these instances are a mix of both continuous and discrete data. There are a number of duplicate or conflicting instances in the dataset the process of cleaning this data will be using drop duplicates method using the code:

dataset.drop\_duplicates()

this has took the row count from 32561 to 32537 removing 24 duplicate rows and has removed 4 rows from the test dataset.

Missing values in the dataset are usually caused by users forgetting to fill in a field, data being lost in the process of creating the dataset, a programming error or users choosing not to fill out a certain field. It is not unusual for this data set to contain missing values as they are more common in datasets containing a large number of fields. It is import these missing values are resolved as absence of information is rarely beneficial to the analysis. In pandas library, I will use the .isnull() method to check both the missing and the NA missing values for each column of the dataset with the code:

print(dataset.isnull())

this produced False across all fields in the dataset showing no missing values in the dataset. This was checked with:

print (dataset.isnull().sum())

to sum all missing data for each column and the results for ever column was 0 indicating no missing values. However in the test dataset there was 1 missing value in all columns except age and using .dropna() method to clean the missing values has removed all missing values from the dataset and the .isnull() method shows 0 missing values on the test dataset. Using this method to remove missing values does remove the entire row of the missing data but due to the size of the dataset the impact of this will be very low. After this using the missingn.matrix() from the missingno library using the code:

>>> missingno.matrix(dataset, figsize=(20,5))

<matplotlib.axes.\_subplots.AxesSubplot object at 0x12A48030>

>>> plt.show()

>>> missingno.matrix(dataset1, figsize=(20,5))

<matplotlib.axes.\_subplots.AxesSubplot object at 0x0FCB4210>

>>> plt.show()

I was able to visualise any missing values in both the dataset and the training dataset left over and the results were:

A picture containing chart

Description automatically generated

Chart

Description automatically generated

Using the missingno visualisation I can see that all columns have solid bars which indicates there are 0 missing values in each column showing the .dropna() function had worked and both datasets have been cleaned of missing data values.

Cleaning all the missing values (?) from the native-country column consisted of replacing all ‘?’ values with the most frequent attribute which would be united states using the code:  
attrib, counts = np.unique(dataset['native-country'], return\_counts = True)

most = attrib[np.argmax(counts, axis = 0)]

dataset['native-country'][dataset['native-country'] == '?'] = most

Summarising the numeric values in the dataset shown for age the mean value is 38 with a standard deviation of 13 which shows a deviation of an observation from the mean. Also the age range varies from 17 to 90 the 1st quartile is 28 with the 3rd quartile being 48 with a difference of 20, however the difference between the 3rd quartile and the maximum is 42 which shows the dater is more dispersed after value 48 and since it is such a big difference it shows the distribution right skewed. The age histogram shows that it is not symmetric and is right-skewed and has 7639 observations after the 3rd quartile and has only 540 values after age 70.

This is similar for fnlwgt as the difference between the minimum and median value is much smaller than the median and maximum value showing a right skew. The histogram shows a right skew since the mean is larger than the median.

For capital-gain it is again rightly skewed as the mean is 1077.6 and median is 0, the 3rd quartile is 0 which shows 75% have 0 capital gain, this shows that a person has 0 gain or a very large gain, capital-gain has the largest standard deviation at 7385 as the spread of values are after the 3rd quartile. The histogram shows that most values are 0 with 29849 of the values, with a few on 10k and 20k also a few on 99k, also shows the concentration on one value with the rest spread with the large standard deviation.

This is similar in capital-loss as up to the 3rd quartile again is 0 capital-loss and the mean is 87 which again shows a right skew but with a lower standard deviation of 402 which shows not as many differ from the median as capital-gain. The histogram shows that again most values are on 0 with 31042 of the values on 0 with a few ranging up to 4k.

The mean value of hours-per week is 40 with the minimum being 1 and the max being 99, the IQR range is between 40 and 45 indicating that 50% of values are between these hours. The values are very sparse below the 1st quartile and above the 3rd quartile showing approximately symmetric data. The histogram shows that most of the values are between 30-40 hours with roughly 17500 of the values also shows that there a few values working over 80 hours per week at 208 values.

The most frequent workclass category is Private with 69.7% (22696) of total values meaning a huge imbalance. The histogram shows there are 8 categories in workclass with private having the highest values, and self-emp-not-inc the second highest with 7.8% of values, the majority of categories have low values (less than 8%) but without-pay and never-worked has the lowest count with less than 1%.

The most biased is native-country with United-States being the most frequent by far with 89.5% (29170) of total values, this is highly biased as native-country also has the highest number of unique categories at 42. The histogram shows an extreme bias toward United-States which is expected as the dataset is from United-States. Race is also highly biased as the white category makes up for 85.42% (27816) of all values. The histogram shows there are 5 unique categories and also it is totally bias towards white and black is the second largest with just 9.59%. The histogram shows there are 5 unique categories and also it is totally bias towards white and black is the second largest with just 9.59%.

Gender is slightly biased towards male making up for 66.9% with just over two thirds of all values at 21790.

Occupation has a much less biased range of categories as compared to other attributes with 15 different unique categories and the highest being prof-specialty with only 12% of total values. The histogram shows that prof-specialty, exec-managerial, craft-repair and adm-clerical sales has similar numbers of values with the lowest being armed-forces with 0.03%.

The distribution of the education histogram shows 16 categories with HS-grad being the most common with the highest value of 32.25% with 10501 of values followed by some-college with 22.39% and bachelors with 16.45% and pre-school with the lowest and least common with 0.26%. the remaining categories all have under 6% of values.

The distribution of the marital status histogram shows 7 unique categories with married-civ-spouse with 45.99% with 14976 values and never-married with 32.81% being the two highest categories and married-af-spouse is the lowest with just 0.07% of values.

The distribution of the relationship histogram shows 6 different unique categories with husband being the highest category with 40.52% and not-in family the second largest with 25.51% and other-relative is the smallest represented category with 3.01% of values.

To determine whether the dataset contains outliers, skewness will be used as several machine learning algorithms assume that the data follows a gaussian distribution, so skewness value makes this easy to check. This was used on the capital-gain column using

print (dataset['capital-gain'].skew())

showing a skewness of: 11.953847687699799 this shows that capital-gain has a right-skewed distribution. Indication presence of extreme higher values. This is shown using dataset[‘capital-gain’].describe() that the max value is 99999. The outliers can be shown visually using boxplots using the code:

plt.boxplot(dataset[“capital-gain”])

plt.show()

*(box plot shown in Appendix B)*

The box plot show that there are indeed outliers at the top of the box plot at the 100000 mark for capital gain, where as the majority are below 50000. Cleaning of these outliers can be done using the trimming techinique by setting suitable minimum and maximum values and drops the index rows from the dataset that do not fall into the min and max values. Using the code:

index = dataset[(dataset['capital-gain']>= 16000)|(dataset['capital-gain'] <= 0)].index

this sets the minimum and maximum values.

dataset.drop(index,inplace=True)

the capital-gain box plot with no outliers shown in *(Appendix B)*

this drops the values that do not meet the min and max values. Using the .describe() function I can see that the outliers have been removed as the new max value of 15831 has shown. With the outliers now removed the column has a skewness of 0.7587292629571624 which comes out between 0 and 1 which is a good skewness rating.

Conducting data analysis on the dataset firstly by identifying all outliers in the dataset by creating box plots for each column in the dataset. The age column boxplot *(see Appendix C)* produced a lot of outliers outside of the maximum of the plot calculated by Q3(75th percentile) + 1.5 \* IQR (Interquartile Range) which is 78 years old so using the index() function the outliers will be removed and the new box plot *(Appendix C)* shows no outliers remaining. The remaining box plots for the columns with continuous data can be shown in *(Appendix D).*

Using the matplotlib library I was able to visualise the relationship between the sex class and the hours-per-week after using sns.countplot() function to visualise the amount of males and females for both the dataset and the test dataset which shown a higher majority of male to female in the adult dataset rather than the test dataset. Then by taking the mean of hours worked for each sex and displaying as a bar chart using the code:  
dataset.groupby ("sex")['hours-per-week'].mean().plot(kind='bar')

*(Result in Appendix E)*

Giving the result that females in the dataset work an average of 39 hours per week and males work an average of 45 hours per week giving a difference of 6 hours. In the test dataset the average hours females worked was 36.4 and males worked an average of 42.4 giving a difference of 5.6 hours. These results slightly differ to the main dataset as both genders work less in total on average, however the difference in hours worked between genders in both datasets remains the same at 6 hours which shows that for both datasets the male gender work 15% more in hours-per-week.

Using the seaborne library the relationship between the class (<=50k or >50k) and the hours-per-week was analysed using a box plot and show <=50k work an average of 38 hours rather than >50k who work 44 hours on average which is an increase of 6 hours. <=50k IQR is much shorter than >50k as it has many more outliers from the minimum to q1 and from q3 to the maximum as >50k IQR is much larger which shows that a much larger

proportion of >50k work more hours than < 50k. Using statistical analysis to test pvalue and ttest the results were

ttest -34.631622855817945

p value 0.0000000000000000000000000000000000000000000000000000

000038990154081891

we reject null hypothesis

which shows that there is a difference in the mean of hours per week and <=50k and >50k and also means that hours-per-week has an effect to differentiate the class.

The boxplot relationship between class and capital-gain shows that most of the capital gains value is at 0 for both classes however <=50k have slightly more outlier values at higher capital-gains above 0 up to 3500. From the t-test statistical analysis the test confirms there is a difference in the mean of <=50k and >50k classes as the results were

ttest -1.4210131221966262

p value 0.1584551469706866

we accept null hypothesis

The capital loss boxplot shown that most values were 0 for both classes with

few outliers.

Using a countplot to show the relationship between the class and the sex shows that male have a much bigger proportion of >50k than female as female have just 2.42% earning >50k, however men also do have a much bigger portion earning <=50k at 41.83% than women at 27% of observations. To determine whether there is a relationship between class and sex using a Pearson’s chi-squared test this test allows us to test whether there is a significant difference between an expected distribution result and an actual distribution result. From the chi-test for sex and class the results were

p\_value 0.2510310778678926

[[23.2 20.8]

[ 5.8 5.2]]

probability=0.950, critical=3.841, stat=1.318

is not dependant

As the results were not dependent it shows us that there is no dependency of sex attribute on the class attribute.

Using multivariate analysis between class, age and sex using a boxplot shows that the median age of females and the median age of males for both classes <=50k and >50k differs minutely with a difference in males having an increase of 3 years, however it shows that the IQR age is larger for males earning >50k showing there are more older males earning this amount. Also females have a slightly older IQR than males earning <=50k.

# Graphical user interface, application, Word Description automatically generatedEvaluation Machine Learning Models

Preprocessing is the first stage in the machine learning workflow, it is considered a highly important step that helps in building machine learning models accurately a general rule in machine learning is the 80/20 rule that data scientists should spend 80% of allotted time for data preprocessing and 20% time to undertake analysis. Data preprocessing consists of cleaning the raw data as the data collected from the real world is converted to the data in a small clean dataset. This is important as real-world data most of the time consists of missing data which can be due to technical issues or when it is not continuously created. Also consists of noisy data such as outliners, this is due to human errors when manually collecting the data or problems with the device used to collect the data. Inconsistent data also occurs in real-world data this can be collected due to human errors either with value names or values or duplication of data. Preprocessing can be performed through various techniques, firstly by conversion of data which consists of converting categorical and ordinal data into numeric features so can be handled by machine learning models. Secondly through ignoring the missing values as when missing data in a data set is found then this row or column can be removed, this is very efficient but should not be performed if there are large amount of missing values present. Thirdly filling missing data in a data set with either the mean, median or most frequent value whenever missing data is encountered. Outlier detection can be used if there is error data present that deviates drastically from other observations in the data set. Finally, machine learning can be used on missing data by predicting what data will be present at the empty position by using existing data.

Learning is the second stage of the machine learning workflow model the goal of this is to research the model that will be the most useful for the type of data using the pre-processed data. Supervised learning model consists of an AI system being presented with data which is labelled, meaning each data is tagged with the correct label. Supervised learning consists of two categories firstly classification problem which is when the variable is categorical as the output can be classified into classes such as blue or red, male or female etc. classification algorithms consist of; K-Nearest Neighbor, Naive Bayes, Decision Trees, Support Vector Machine and Logistic Regression.

The second supervised learning category is Regression problem this is when the target variable is continuous as the output is numeric. Regression algorithms consists; Linear regression, support vector regression, decision trees, Gaussian progresses regression and Ensemble methods. Unsupervised learning when an AI system is presented with unlabeled, un-categorised data and the systems algorithms act on the data without previous training. The output is then depended on the algorithms that have been coded. This is categorised into clustering and association. Clustering consists of a set of inputs divided into groups, however the groups are not known before hand unlike classification. Methods used for clustering; Gaussian mixtures, K-means Clustering, Boosting, Hierarchical Clustering.

Model Evaluation is the third stage of machine learning workflow, this is a highly important stage of the development process as it aids in finding the most useful model that represents the data and how effective the model will work in future uses.

Then finally is the prediction stage to improve the model the hyper-parameters of the model can be tuned to try to improve accuracy. Also using the confusion matrix to try to increase the number of true positives and true negatives.

k-Nearest neighbor (KNN) algorithm is a supervised machine learning algorithm used to solve both classification and regression problems. It relies on labeled input data to learn a function that produces an output when given new unlabeled data.

Advantages of k-Nearest neighbor are:

* The algorithm is simple and easy to implement.
* There is no need to tune parameters, make any additional assumptions, or build a model.
* Very versatile algorithm as can be used for classification and regression.

The disadvantages of KNN:

* The algorithm gets significantly slower as the number of examples and or independent variables increase.
* There are faster algorithms that can produce more accurate classification and regression results.
* Impractical choice where predictions are needed in a short time frame.

The parameters of KNN consist of:

* n\_neighhbors: int and default=6, this is the number of neighbors to use by default for kneighbors queries.]
* weights: {‘uniform’,’distance’} or callable, default = ‘uniform’, weight function used in prediction.
* Algorithm: {‘auto’,’ball\_tree’,’kd\_tree’,’brute’}, default = ‘auto’, algorithms used to compute the nearest neighbors.
* Leaf\_size: int, default = 30, leaf size passed to balltree or kdtree.
* P: int, default=2, power parameter for the Minkowski metric.
* Metric: str or callable, default=’minkowski’, the distance metric to use for the tree.
* Metric\_params: dict, default = None, additional keyword arguments for metric function.
* n\_jobs: int, default= None, number of parallel jobs to run for neighbors search.

KNN consists of range differences as with KNN measure distances assume that all variables have the same range of value, however with real data this is likely not the case, so the ranges need to be standardised through normalisation, this standardises a set of numbers so each value is between 0.0 and 1.0, the minimum observed value is subtracted by each number in turn and then divided by the observed range of values. This is also the case for any algorithm where distance plays a vital role for prediction or classification. If the data is not numeric when performing KNN such as categorical data then sparsification can be used by creating a new binary 0/1 variable to represent every non-numeric value of every original variable to create a new, continous dataset.

Performing KNN on the adult dataset firstly consisted of transforming all categorical data into numerical integers through using the LabelEncoder of sklearn using the code:

Labelencod\_workclass = LabelEncoder()

dataset['workclass'] = Labelencod\_workclass.fit\_transform(dataset['workclass'])

Using this code on all categorical columns in the dataset to convert all to integers in order to carry out KNN. Then the values were scaled using standardscaler from the sklearn library using the code:

standard\_scaler = StandardScaler()

standard\_scaler.fit(dataset.drop('class',axis=1))

stanscaled\_feat = standard\_scaler.transform(dataset.drop('class',axis=1))

The x and y train test splits were created for the model and declaring the test size using the train\_test\_split library from sklearn. Then the KNN model is created using KneighborsClassifier and the number of neighbors are declared, then using y\_pred it predicts the values. A classification report was then produced as well as the confusion matrix results. Then using the results from the table to visually see where the downward curve flattens to then find the best K value to use which came out at 13 so the new n\_neigbors parameter was 13 to create the most accurate results for Kneighbors.

The results can be found in chapter 4.

Random Forest algorithm is a machine learning algorithm which is used for both classification and regression. The forest is made up of trees and since it uses a lot of trees it is a very robust algorithm that that works in four steps:

1. select random samples from a given dataset
2. constructs decision trees for each sample and gets a prediction result for each tree
3. performs a vote for each predicted result
4. selects the prediction result with the most votes as the final prediction

When conducting the random forest algorithm on the dataset, firstly the RandomForestClassifier is imported from the sklearn.enemble library. Then creating a randomforestclass and fitting with the X and Y training sets using code:

randomforest = RandomForestClassifier()

randomforest.fit(X\_train, y\_train)

then storing the score from the randomForest model into the score class and displaing sing Print to get the 0.91 91% accuracy result, then printing the confusion matrix for the model and creating a heat map using the matplotlib library to virtually display the results for the model using the code:

plt.figure(figsize=(9,9))

sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues\_r');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all\_sample\_title = 'Accuracy Score: {0}'.format(score)

plt.title(all\_sample\_title, size = 15);

The results can be found in chapter 4.

The advantages of Random Forest algorithm are:

* Random forests are considered highly accurate due to the number of decision trees in the process
* Does not get affected by overfitting since it takes the average of all predictions cancelling out the biases.
* Algorithm can be used in both classification and regression
* Algorithm can handle missing values, either using median values to replace continuous variables or compute the proximity-weighted average of missing values.

The disadvantages are:

* Being slow in generating predictions since it has multiple decision trees so it is very time-consuming
* The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following path in tree.

Parameters for Random Forest:

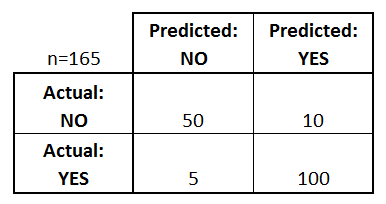
* N\_estimators: int, default=100, the number of trees in the forest
* Criterion: {“gini”,”entropy”}, default = “gini”, the function to measure the quality of a split.
* Max\_depth: int, default = None, the maximum depth of the trees
* Min\_samples\_split: int or float, default = 2, the minimum number of samples required to split an internal node.
* Min\_samples\_leaf: int or float, default=1, the minimum number of samples required to be at a leaf node.
* Min\_weight\_fraction\_leaf: float, default= 0.0, the minimum weighted fraction of the sum total of weights required to be at a leaf node.
* Max\_features:{“auto”, “sqrt”, “log2”}, int or float, default=”auto”, the number of features to consider when looking for the best split.
* Max\_leaf\_nodes: int, default=None, grow trees with max\_leaf\_nodes in best-first fashion. Best nodes are defined as relative reduction in impurity.
* Min\_impurity\_decrease: float, default=0.0, a node will be split if this split induces a decrease of the impurity greater than or equal to this value.
* Min\_impurity\_split: float, default=None, threshold for early stopping in tree growth
* Bootstrap: bool, default=True, whether bootstrap samples are used when building trees.
* Oob\_score:bool, default= False, whether to use out-of-bag samples to estimate the generalisation accuracy
* N\_jobs: int, default=None, the number of jobs to run I parallel
* Random\_state: int or RandomState, default=”None”, controls both the randomness of the bootstrapping of the samples used when building trees.
* Verbose: int, default=0, controls the verbosity when fitting and predicting.
* Warm\_start:bool, default=False, when set to true
* Class\_weight:{“balanced”, “balanced\_subsample”}, dict or list of dicts, default=None
* Ccp\_alpha: non-negative float, default = 0.0, Complexity parameter used for Minimal cost-complexity pruning
* Max\_samples: int or float, default=None, if bootstrap is true, the number of samples to draw from X to train each base estimator.

Logistic regression is a machine learning algorithm used for classification problems, it is a predictive analysis algorithm based on concept of probability. Types of logistic regression are binary and multi-linear functions failsClass. Scaling/normalisation can be good for linear regression so that a large numerical value does not overwhelm a smaller value. This allows weights learnt for the variable to be within a tighter range.

When performing logistic regression on the data set firstly logistic regression was imported from the sklearn.linear\_model library and a class was created to hold both x and y test data using: logisticRegr.fit(X\_train, y\_train) and using the score function to produce a logistical regressin score: score = logisticRegr.score(X\_test, y\_test)

print(score)

this created the accuracy score for the data set and also using metric.confusion\_matrix to produce the confusion matrix. Following this a heatmap was created to visually display the logistic regression results using the accuracy score and the confusion matrix. Confusion matrix example:



The results can be found in chapter 4.

Advantages of logistic regression:

* Convenient probability scores for observations
* Efficient in implementations

Disadvantages:

* Does not perform well when feature space is too large
* Does not handle large numbers of categorical features/variables well.
* Relies on transformations for non-linear features.

Parameters for Logistic Regression:

* Penalty: {‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default= ‘l2’, used to specify the norm used in the penalisation.
* Dual: bool, default= False, dual, or primal formulation, dual is only implemented for l2 penalty with liblinear solver.
* Tol: float, default = 1e-4, tolerance for stopping criteria
* C: float, default= 1.0, inverse of regularization strength, a positive float
* Fit\_intercept: bool, default=True, specifies if a constant (i.e. intercept) should be added to the decision function
* Intercept\_scaling: float, default=1, useful only when the solver ‘liblinear’ is used and self.fit\_intercept is set to True.
* Class\_weight: dict or ‘balanced’, default = None, weights associated with classes in the form {class\_label:weight}. If not given, all classes are supposed to have weight one.
* Random\_state: int, RandomState instance, default=None, used when solver == ‘sag’, ‘saga’ or ‘liblinear’ to shuffle the data.
* Solver: {‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’}, default=’lbfgs’, alorithm to use in the optimisation problem, for small datasets, liblinear is a good choice.
* Max\_iter: int, default = 100, maximum number of interations taken for the solvers to converge.
* Multi\_class: {‘auto’, ‘ovr’, ‘multinomial’}, default = ‘auto’, if the option chosen is ‘ovr’, then a binary problem is fit for each label
* Verbose: int, default = 0, for the liblinear and lbfgs solvers set verbose to any positive number for verbosity.
* Warm\_start: bool, default= False, when set to true, reuse the solution of the previous call to fit as initialisation, otherwise, erase the previous solution.
* N\_jobs: int, default=None, number of CPU cores used when parallelising over classes if multi\_class=’ovr’. This parameter is ignored when the solver is set to ‘liblinear’ regardless of whether ‘multi\_class’ is specified or not.
* L1\_ratio: float, default= None, the elastic-net mixing parameter, with 0<=l1\_ratio<=1.

# Evaluation Machine Learning Models

**k-Nearest neighbor (KNN)**

The results for K nearest neighbor were

Graphical user interface, application

Description automatically generated

The graph shows that after value 13 the reduce rate flattens and does not reduce much so taking the value k=13 to produce the classification report and confusion matrix. The new report results were:

precision recall f1-score support

0 0.88 0.91 0.89 12243

1 0.68 0.58 0.63 3818

accuracy 0.84 16061

macro avg 0.78 0.75 0.76 16061

weighted avg 0.83 0.84 0.83 16061

The reports show the main classification metrics precision. The metrics are calculated by using true and false positives, true and false negatives. The recall is the ability of a classifier to find all positive instances. For each instances classified positive, what percentage were correct. The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. The support is the number of actual occurrences of the class in the specified dataset, imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or rebalancing.

Confusion matrix result score:

[[11193 1050]

[ 1585 2233]]

This shows that the precision has increased for both 0 and 1 as 1 increased by 0.01 and 1 increased by 0.17 which is a big increase also the accuracy score went from 0.80 to 0.83. the confusion matrix shown a 3283 predicted but an actual of 3818. Which is quite a large difference.

**Logistic Regression**

The accuracy score for logistic regression was 0.8270344312309321 which is an 83% accuracy. The classification report is:

precision recall f1-score support

0 0.85 0.94 0.89 7315

1 0.72 0.46 0.56 2322

accuracy 0.83 9637

macro avg 0.78 0.70 0.73 9637

weighted avg 0.81 0.83 0.81 9637

The confusion matrix results shown a 2508 predicted however with a 3818 which is a large difference.

[[11509 734]

[ 2044 1774]]

Graphical user interface, application

Description automatically generatedThe heatmap for the accuracy:

# 

The heatmap shows the accuracy score and the confusion matrix visually in order to make it easier to analyse all the results.

**Random Forest**

The accuracy score for the random forest algorithm was 0.9178167479506071 or 92%. The classification report results are:

precision recall f1-score support

0 0.93 0.97 0.95 7315

1 0.88 0.77 0.82 2322

accuracy 0.92 9637

macro avg 0.90 0.87 0.88 9637

weighted avg 0.92 0.92 0.92 9637

The confusion matrix results show a 2030 predicted with a 2322 actual which is the most accurate out of all three models for the confuion matrix.

[[7061 254]

[ 538 1784]]

The heatmap for the accuracy:

Graphical user interface, application

Description automatically generated

# Discussions

After evaluating the findings from the three machine learning models I can conclude that the random forest algorithm is the most accurate model for this dataset and to solve the problem as to predict whether income exceeds 50K per year based on census data. . The random forest score was the highest with 91.7% which is a high accuracy score, when compared to the accuracy score of k-nearest neighbour with a score of 83% and logistical regression with 82% showing KNN and logistical regreen produced similar results, were as random forest was 9% more accurate. This is also evident in the confusion matrix as the random forest had the lowest difference between predicted result and actual result. This was most likely due to the fact random forest uses many decision trees to calculate the accuracy.

During the development of this report I have gained an increased understanding into a range of different machine learning methods for data analytics and also a range of exploratory data analysis methods, such as cleaning data and analysing relationships using different charts and plots. Also, after analysing different algorithms such as Random Forest, Linear regression and KNN I have gained a greater understanding into the role these algorithms play in machine learning in a wider context.

Overall, this module has helped my understanding of machine learning for data analytics as it has increased my Python programming skills as I have used a range of libraries and functions throughout the project. Also, by expanding my depth of understanding of machine learning models and by using a real-world example in the assignment has helped me understand how machine learning datasets are used and analysed in real-world applications.

# Bibliography

UCI Machine Learning Repository, 1996. *Adult Data Set*, viewed 21 Jan 2020, <<https://archive.ics.uci.edu/ml/datasets/adult>>

*Ibm.com. 2020. IBM Knowledge Center. [online] Available at: <https://www.ibm.com/support/knowledgecenter/SS3RA7\_15.0.0/com.ibm.spss.crispdm.help/crisp\_overview.htm> [Accessed 30 April 2020]. Medium. 2020. Ways To Detect And Remove The Outliers. [online] Available at: <https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba> [Accessed 5 May 2020]. Kaggle.com. 2020. Simple EDA For Beginners. [online] Available at: <https://www.kaggle.com/alokevil/simple-eda-for-beginners> [Accessed 14 May 2020]. Pant, A., 2020. Workflow Of A Machine Learning Project. [online] Medium. Available at: <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94> [Accessed 14 May 2020].*

*Medium. 2020. Introduction To Logistic Regression. [online] Available at: <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148> [Accessed 14 May 2020].*

*Medium. 2020. Machine Learning Basics With The K-Nearest Neighbors Algorithm. [online] Available at: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761> [Accessed 14 May 2020].*

*Sachan, L. and Sachan, L., 2020. Logistic Regression Vs Decision Trees Vs SVM: Part II- Edvancer Eduventures. [online] Edvancer.in. Available at: <https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2/> [Accessed 14 May 2020].*

*Scikit-learn.org. 2020. Sklearn.Linear\_Model.Logisticregression — Scikit-Learn 0.23.0 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html> [Accessed 14 May 2020].*

*Scikit-learn.org. 2020. Sklearn.Neighbors.Kneighborsclassifier — Scikit-Learn 0.23.0 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html> [Accessed 14 May 2020].*

*Scikit-learn.org. 2020. 3.2.4.3.1. Sklearn.Ensemble.Randomforestclassifier — Scikit-Learn 0.23.0 Documentation. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> [Accessed 14 May 2020].*

*Tutorialspoint.com. 2020. Classification Algorithms - Random Forest - Tutorialspoint. [online] Available at: <https://www.tutorialspoint.com/machine\_learning\_with\_python/machine\_learning\_with\_python\_classification\_algorithms\_random\_forest.htm> [Accessed 14 May 2020].*

*Data School. 2020. Simple Guide To Confusion Matrix Terminology. [online] Available at: <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/> [Accessed 14 May 2020].*

*Kaggle.com. 2020. Simple EDA For Beginners. [online] Available at: <https://www.kaggle.com/alokevil/simple-eda-for-beginners> [Accessed 14 May 2020].*

*Medium. 2020. Introduction To Logistic Regression. [online] Available at: <https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148> [Accessed 14 May 2020].*

*Medium. 2020. Logistic Regression Using Python (Scikit-Learn). [online] Available at: <https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a> [Accessed 14 May 2020].*

*Sachan, L. and Sachan, L., 2020. Logistic Regression Vs Decision Trees Vs SVM: Part II- Edvancer Eduventures. [online] Edvancer.in. Available at: <https://www.edvancer.in/logistic-regression-vs-decision-trees-vs-svm-part2/> [Accessed 14 May 2020].*

*Scikit-yb.org. 2020. Classification Report — Yellowbrick V1.1 Documentation. [online] Available at: <https://www.scikit-yb.org/en/latest/api/classifier/classification\_report.html> [Accessed 14 May 2020].*

*Users.sussex.ac.uk. 2020. Machine Learning - Lecture 2: Nearest-Neighbour Methods. [online] Available at: <http://users.sussex.ac.uk/~christ/crs/ml/lec01b.html> [Accessed 14 May 2020].*

*AJCA (2020). Income data using KNN. [online] kaggle.com. Available at: https://www.kaggle.com/abhijeetjainmca/income-data-using-knn [Accessed 14 May 2020].*

*Bourke, D. (2018). Code with me (live): How to make your first Kaggle submission from scratch! YouTube. Available at: https://www.youtube.com/watch?time\_continue=1008&v=f1y9wDDxWnA&feature=emb\_logo [Accessed 14 May 2020].*

# Appendix

*Appendix A*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Adult Training Dataset | | | | Adult Test Dataset | | | |
| Attribute | Attribute Description | Data type | Min | Max | Avg | Std Dev | Min | Max | Avg | Std Dev |
| Age | Age of each participant in the census. Stored as Integer. | Int64 | 17 | 90 | 38.58 | 13.64 |  |  |  |  |
| Workclass | The industry the participant is employed by e.g. Private, Federal-gov. | Object |  |  |  |  |  |  |  |  |
| Fnlwgt |  | Int64 | 1.0555 | 1.4847 | 1.8978 | 1.055 | 1.349 | 1.49 | 1.894 | 1.057 |
| Education | The highest level of education awarded to the participant e.g bachelors | Object |  |  |  |  |  |  |  |  |
| Education-num |  | Int64 | 1.00 | 16.0 | 10.08 | 2.57 | 1.00 | 16.00 | 10.07 | 2.567 |
| Marital-status | Distinct options that describes a participant’s legal relationship status | Object |  |  |  |  |  |  |  |  |
| Occupation | The job role the participant works in e.g. tech-support, craft repairer. | Object |  |  |  |  |  |  |  |  |
| Relationship | The relationship status of the participant e.g. wife, own-child, Husband. | Object |  |  |  |  |  |  |  |  |
| Race | The race of each participant in the census e.g. white, Asian-Pac-islander. | Object |  |  |  |  |  |  |  |  |
| Sex | The gender of each participant in the census. | Object |  |  |  |  |  |  |  |  |
| Capital-gain | Capital-gain is the gain in capital each participant has gained per annum | Int64 | 0.00 | 99999.00 | 1077.65 | 7385.29 | 0.00 | 99999.0 | 1081.9 | 7583.9 |
| Capital-loss | Capital-loss is the loss in capital each participant has decreased per annum. | Int64 | 0.00 | 4356.00 | 87.3 | 402.96 | 0.00 | 3770.0 | 87.899 | 403.1 |
| Hours-per-week | The number of hours per week each participant works | Int64 | 1.00 | 99.00 | 40.43 | 12.347 | 1.00 | 99.00 | 40.39 | 12.479 |
| Native-country | The country where the participant was born or native too. | Object |  |  |  |  |  |  |  |  |
| class | Whether the participant earnt less than or greater than $50k per annum. | Object |  |  |  |  |  |  |  |  |

Graphical user interface, application, Word

Description automatically generated*Appendix B- capital gain*

Graphical user interface, chart

Description automatically generated

Appendix C-

Graphical user interface

Description automatically generatedChart, box and whisker chart

Description automatically generated

Graphical user interface, application, Word

Description automatically generatedGraphical user interface, application, Word

Description automatically generatedAppendix E- Dataset and test dataset comparison