Predict the income of adult from dataset of census income

Yujie Zhang, Master of Professional Study

University of Auckland

Abstract

With the improvement of monetary world broadly, SDG distributed the objective of advancing supportable financial development and gainful work for all in the world. The most effective method to accomplish more elevated amounts of profitability in different regions by giving all the more nice activity is one of the objectives for this objective. The article targets foreseeing the salary of a grown-up as per the aftereffect of machine learning on the dataset of registration pay. The strategy for the looking into incorporates a few stages like business understanding, information understanding, information planning, information mining technique choice, information mining calculation determination, information mining, elucidation. Python is chosen as the product suite for this undertaking. Ends and further work are given at last after the examination as indicated by the aftereffect of break down.

Keywords: Machine Learning, python

Business understanding

Objectives of the business

Sustainable Development Goals can be implemented by means which are generated by the growth of inclusive and sustainable economic growth which is also drive progress of it. Nowadays the problem of relatively low productivity and high level of unemployment still can not be resolved. Meanwhile, the global economy is growing at a slower rate. To increase the opportunities of employment, especially for young peopleMore progress is needed to increase opportunities of employment, reduce employment which is informal with the pay gap of gender and also secure the environments of working for decent work, more progress is needed. The motivations of this project are to analyse the dataset to find the most important factors which are able to effect and predict the income of adult.

The objectives of this project is to answer the following questions:

What is the most important feature which affect the income of adult?

What is the relationship between income of adult with the other features?

How to predict if the income of adult would be larger than 50K or less than it?

Which is the best model to do the prediction and the accuracy of it?

Assess the situation

Now that the goals of this project have already been defined, it's time to access the situation of the project. In this step, I will describe the resource which include data and personnel. Also the risk of the project is necessary to discuss with a contingency plan for each risk.

Personnel. It's clear that the researcher(me) is the only human resource for this project. The resource of this lecture and the laboritory is of much help for me to have a good start on project. Tutors, who are data mining specialist with much experience answer questions of the project during laborility and office time.

Data. There is plenty of data to draw from. In fact, for this initial study, the project will restrict the analysis to the information of adults who has records from census database. The “adult” datasets which are available on UCI Machine Learning Repository are selected for this project. The dataset was extracted by Barry Becker from the 1994 Census database, and it is popular and widely used for researching in different projects.

Risks. Aside from the time spent by the researcher on the study, there is no serious risk for this project. However, time is always important because machine learning is always time consuming, so this project is scheduled for a whole week in this semester. Hardware maybe another problem of this project. Building the different models may cost much memory of the computer, so the hardware situation maybe not enough to support this since the experiments are conducted in my laptop. If this situations exists, then some scopes of the project maybe rearranged. Also, technical problem is another risk for this project, but online resource from internet may be able to help to resolve the technical issue when executing different models.

Data mining objectives

After assessing the business situation of the project, to define a technical solution to this business problem, all things need to be kept concrete. So the business objectives of the project need be translated into data mining term. The goals for the initial study to be completed are:

- Use historical information about previous income of adults to generate multiple models of classification to perform comparisions between different models and find the pattern of the data. Also, the importance and relationship between factors would be provided by this project.
- Select correct method and algorithm to predict if the income for an adult would be larger or less than 50K per year.
- The evaluation of the model will be performed in different way including logarithms loss, confusion matrix and classification report. The model with high performance will be selected as the best model in the end. The basic line to select the model is that the accuaracy is larger than 90% and the f1 score of the model is larger than 0.8.

Project plan

After setting the practical and researching objectives, the plan of the research project is generated. Table 1 shows the details of the plan.

Phase	Time	Risks
Business understanding	1 day	No
Data Understading	1 day	Data problems, technology problems
Data preparation	1 day	Data problems, technology problems
Modeling	2 days	Technology problems, inability to find adequate model
Evaluation	1 day	Inability to implement results
Deployment	1 day	Inability to implement results

Table 1. Project Plan

Data Understanding

In this phase, I will continue to take a closer look at the data available for mining. It is critical to have a good understanding of data and avoid protential problems during data preparation. In this section, I will access the data and explore it using tables and graphics. This makes it possible for me to determine the quality of the data and describe the results of these steps in the project documentation.

Collect initial data

The dataset in this project uses only one data source:

Existing Data. The data is available from UCI Machine Learning Repository. It contains part of the records of Census database in 1994 and is extracted by certain rules. The dataset contains all of the information of adult including income and other information like education and occupation. The format of the dataset is csv, and size of the dataset is also able to meet the requirement of this project. There are some attributes will be considered to be useless in the following steps..

Describe the data

The descriptions of dataset in this project are focusing on the quantity and quality of the data. Listed below are some key characteristics of the dataset.

Amount of data. For most techniques of models, there are trade-offs correspond with data size. Large volume of data sets can generate models which are more accurate, however they can also increase the length of the processing time. In this project, there are totally 15 fields in the table with 48842 records in the dataset.

Meaning of data attributes. The meaning of each field of the dataset is as following:

age: the age of adult.

workclass: the workclass of an individual including Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: sampling weighting.

education: the education level of an individual, including Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: the number of education of an individual, the numeric format of education

marital-status: the marital status of an individual including the Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: the occupation of an individual including Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: the relationship of family for an individual including Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: the race of an individual including White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: the capital gain of an individual

capital-loss: the capital loss of an individual

hours-per-week: the number of working hours for an individual each week

native-country: the native-country of an individual including United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-

Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Value types. The describe function of pandas package is used to describe the continuous variables of dataset. According to the result from Figure1, there are 6 continuous variables type which are age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week. The others are all categorical variables. According to the result, the average age is about 38. Figure1 shows the code used to describe the data and the result.

```
path = '.'
all_files = glob.glob(path + "/*.csv")
li = []

for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)

print(frame.describe())
```

```
                age        fnlwgt  education-num  capital-gain  capital-loss
count  48842.000000  4.884200e+04   48842.000000  48842.000000  48842.000000
mean      38.643585  1.896641e+05      10.078089   1079.067626     87.502314
std       13.710510  1.056040e+05       2.570973   7452.019058    403.004552
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.175505e+05       9.000000      0.000000      0.000000
50%       37.000000  1.781445e+05      10.000000      0.000000      0.000000
75%       48.000000  2.376420e+05      12.000000      0.000000      0.000000
max       90.000000  1.490400e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    48842.000000
mean        40.422382
std         12.391444
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

Figure 1. Description of continuous attribute

Explore the data

To address the data mining goal, I explored the dataset using charts and visualisation tools including plotting the density and counting of raw data and applying basic statistic method. This can also help to formulate the hypotheses of the data and shape the data transformation tasks which will take place during the step of data preparation. Figure 2 shows the code which is used to create density plot.

```
f, (ax1) = mp.subplots(figsize=(8, 8))
ax1 = sns.distplot(age, rug=True, rug_kws={"color": "g"},
                   kde_kws={"color": "k", "lw": 3, "label": "KDE of age"},
                   hist_kws={"histtype": "step", "linewidth": 3,
                             "alpha": 1, "color": "g", "label":"Histgram of age"})


f, (ax3) = mp.subplots(figsize=(8, 8))
ax3 = sns.distplot(capital_gain, rug=True, rug_kws={"color": "g"},
                   kde_kws={"color": "k", "lw": 3, "label": "KDE of capital_gain"},
                   hist_kws={"histtype": "step", "linewidth": 3,
                             "alpha": 1, "color": "g", "label":"Histgram of capital_gain"})

f, (ax4) = mp.subplots(figsize=(8, 8))
ax4 = sns.distplot(capital_loss, rug=True, rug_kws={"color": "g"},
                   kde_kws={"color": "k", "lw": 3, "label": "KDE of capital_loss"},
                   hist_kws={"histtype": "step", "linewidth": 3,
                             "alpha": 1, "color": "g", "label":"Histgram of capital_loss"})

f, (ax5) = mp.subplots(figsize=(8, 8))
ax5 = sns.distplot(hours_per_week, rug=True, rug_kws={"color": "g"},
                   kde_kws={"color": "k", "lw": 3, "label": "KDE of hours_per_week"},
                   hist_kws={"histtype": "step", "linewidth": 3,
                             "alpha": 1, "color": "g", "label":"Histgram of hours_per_week"})
```

Figure 2. Code of Density plot

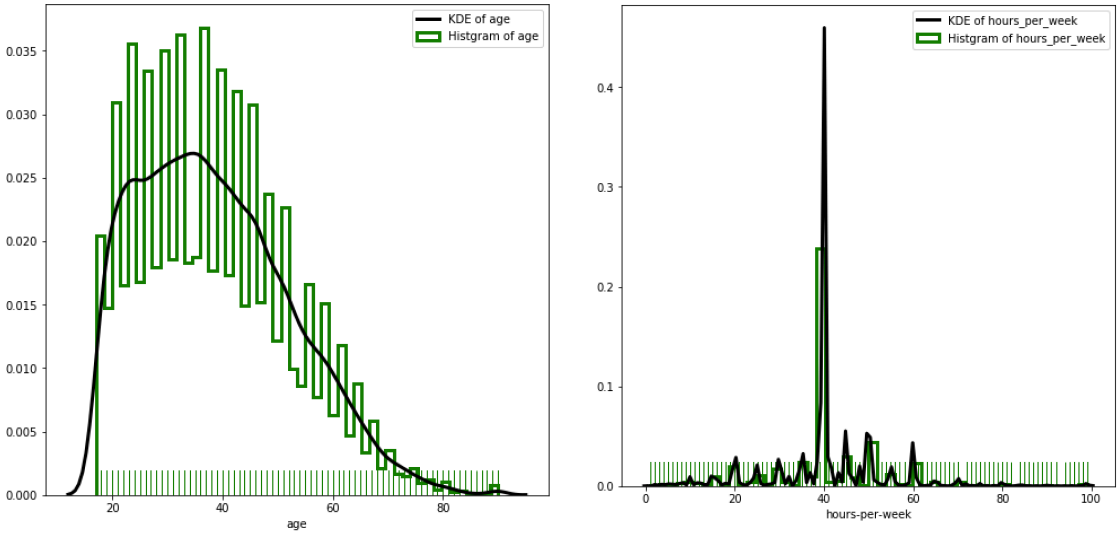


Figure 3. Density plot of age and hours-per-week

Figure 3 shows the density graph of age and hours-per-week. The graph shows that the majority of age adult in dataset is between 18 to 60 years old. For adults of above 60 years old, the data may be able to be removed since they are almost at retire age. According to the graph, the working hours of adult is mostly about 40 hours, and this is reasonable because the average working hour is eight hour per day with five official working dat per week. More analysis will be conducted during the data preparation step.

Figure 4 shows the density graph of capital_gain and capital_loss. The graph shows that the majority of adult in dataset have zero values of capital_gain and capital_loss, so we can drop these two columns in the dataset.

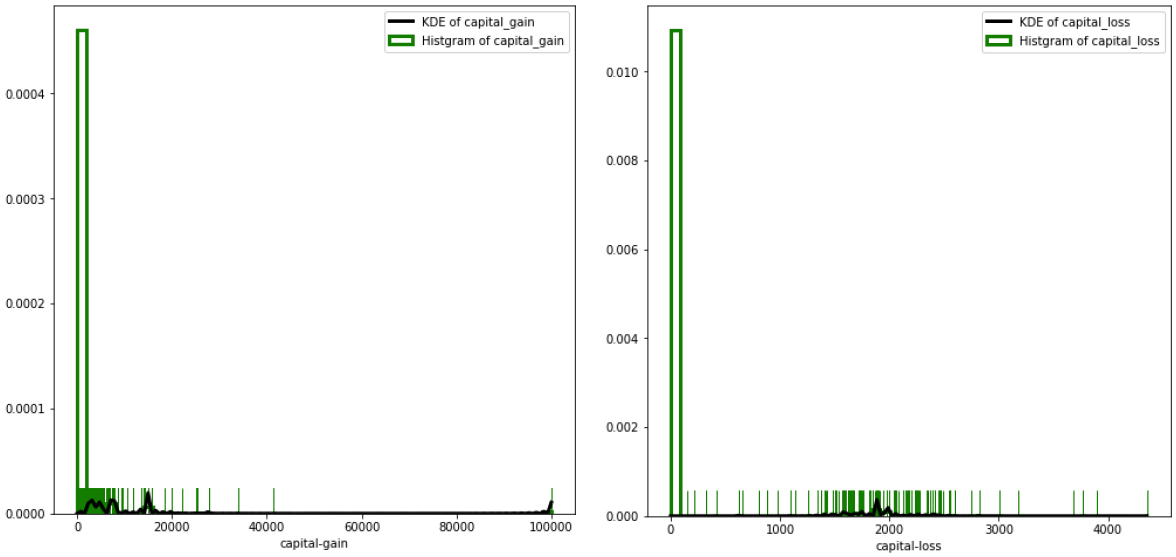


Figure 4. Density plot of capital_gain and capital_loss

Figure 5 shows the distribution of education and marital-status. According to the above graph, there are three categories of education which are of higher percentage than others including Bachelor,HS-grad and Some-college. The percentage of them is 16.4%, 32.2% and 22.3% respectively. Other categories of education are of lower percentage which are below 5%. The graph also shows that there are 45.8% of adults are of Married-civ-spouse category. About 33% of the adults who have never been married and 13.6% of them are divorced. Other categories of them are of lower percentage which is less than 5%.

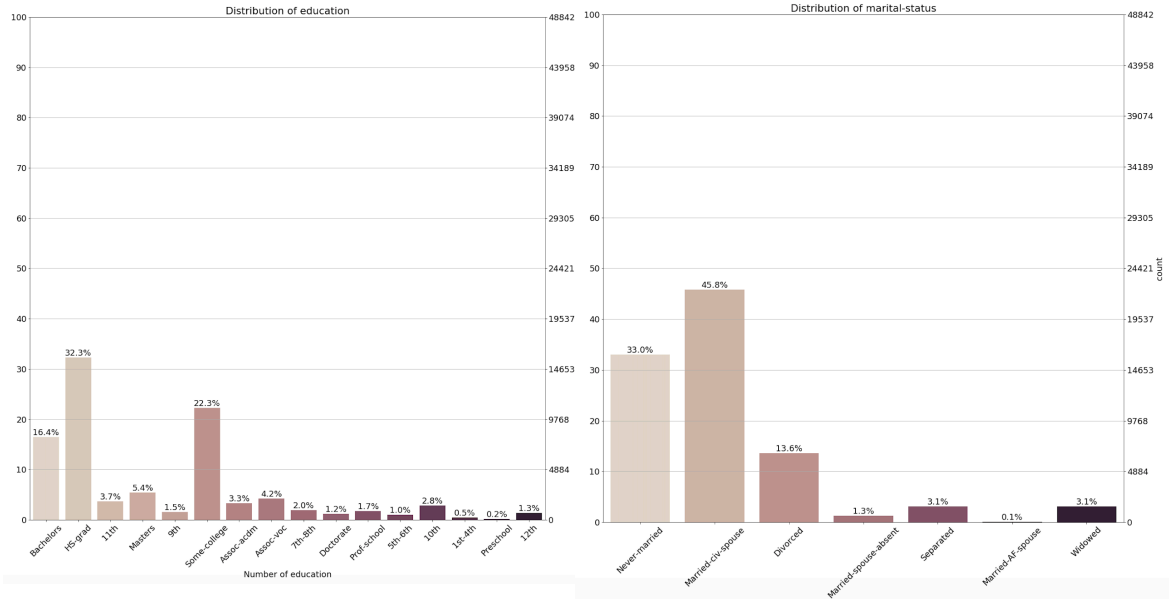


Figure 5. Distribution of education and marital-status

Figure 6 shows the distribution of race and native-country. According to the above graph, the percentage of white people is much higher than other categories which is about 85.5%. The percentage of black people is about 9.6%. The percentage of people in other race is all lower than 5%. The graph also shows that the majority of adults are from United States, and the percentage reached to about 89.7%. This also indicated that the native-country maybe not so important in this project. It's possible to consider to drop this column in the following steps.

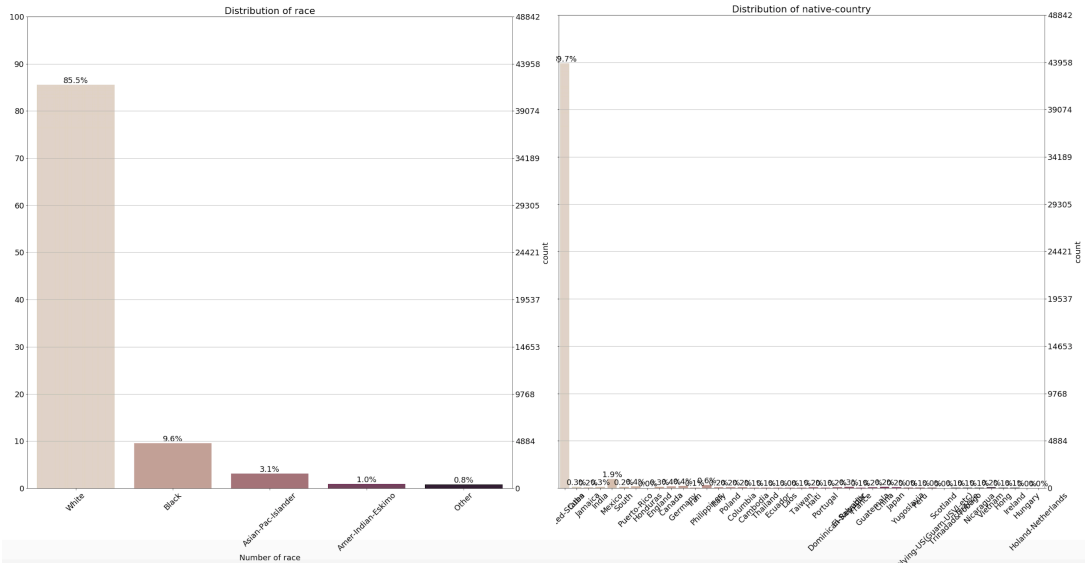


Figure 6. Distribution of race and native-country

Figure 7 shows the distribution of work-class and occupation. According to the above graph, the majority of adults is doing private business and the percentage of it is about 69.4%. For other types of work class like state-gov, self-emp-not-inc and so on, the percentage of them are much lower. The graph also shows the distribution of occupation which looks more balanced. The percentages of most occupations are around 10%. There are some categories like protective-serv and priv-house-serv, of which the percentage is almost 0.

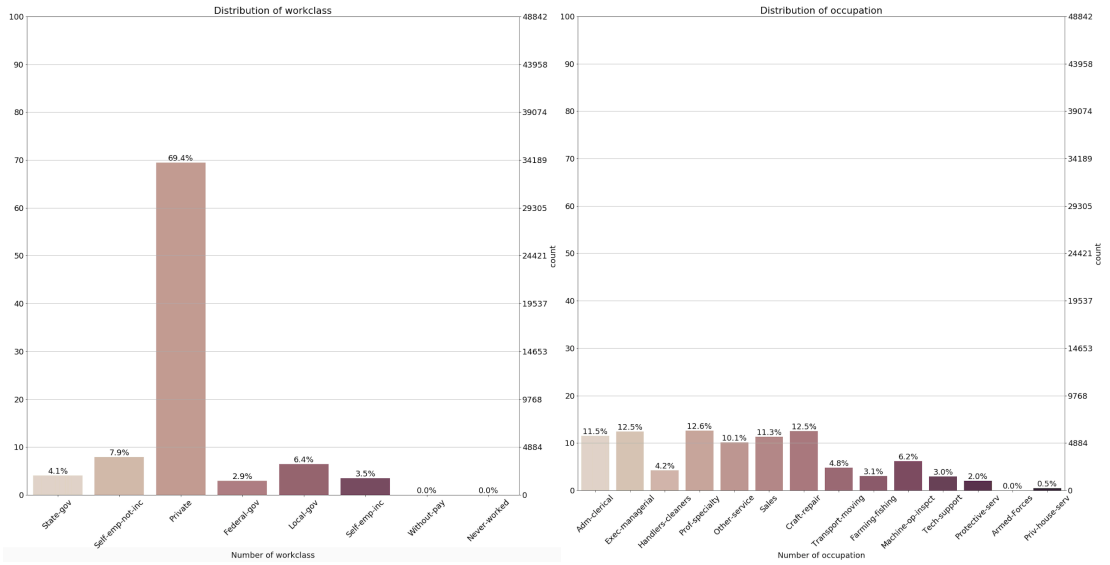


Figure 7. Distribution of work-class and occupation

Figure 8 shows the distribution of relationship in family and the income of adult. The graph indicates that there are about 40 percentage of adults are acting as husband and there are about 25 percentage of adults are not in a family. The percentage of adults who owns a child is about 15.5%. There are about 10 percentage of adults who are not married. The graph of income indicates that there are about 76 percentage of adults of which the income is less than 50K per year, and the rest of them are larger than 50K per year.More analysis will be conducted during the data preparation step.

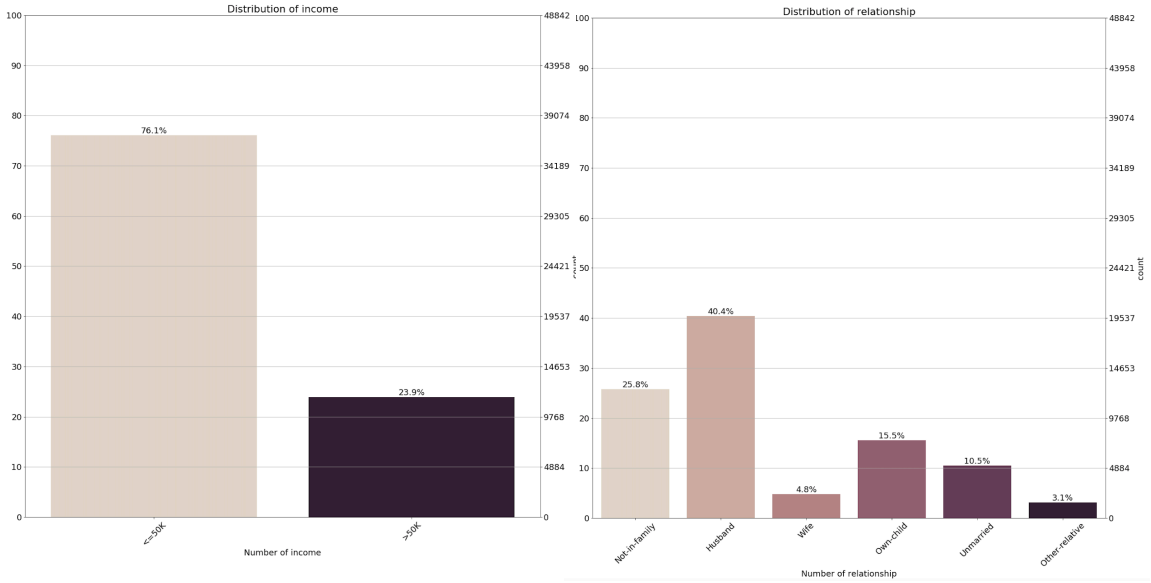


Figure 8. Distribution of relationship and income

Figure 9 shows the code which is used to create above distribution plot.

```
for cal in cat_col_list:
    mp.figure(figsize=(20,20))
    ax7 = sns.countplot(x=cal, data=frame, palette="ch:.25")
    mp.title('Distribution of ' + cal)
    mp.xlabel('Number of ' + cal)
    mp.xticks(rotation=45)

    # Make twin axis
    ax8=ax7.twinx()

    # Switch so count axis is on right, frequency on left
    ax8.yaxis.tick_left()
    ax7.yaxis.tick_right()

    # Also switch the labels over
    ax7.yaxis.set_label_position('right')
    ax8.yaxis.set_label_position('left')

    ax8.set_ylabel('Frequency [%]')


    for p in ax7.patches:
        x=p.get_bbox().get_points()[:,0]
        y=p.get_bbox().get_points()[1,1]
        ax7.annotate('{:.1f}%'.format(100.*y/len(frame[cal])), (x.mean(), y), ha='center', va='bottom')
    # Use a LinearLocator to ensure the correct number of ticks
    ax7.yaxis.set_major_locator(ticker.LinearLocator(11))

    # Fix the frequency range to 0-100
    ax8.set_ylim(0,100)
    ax7.set_ylim(0,len(frame[cal]))

    # And use a MultipleLocator to ensure a tick spacing of 10
    ax8.yaxis.set_major_locator(ticker.MultipleLocator(10))

    # Need to turn the grid on ax2 off, otherwise the gridlines end up on top of the bars
    ax8.grid(None)

    mp.rcParams.update({'font.size': 18})

    mp.savefig('/Users/yujzhang/Documents/722_plot/plot'+cal+'.png', format='png')
```

Figure 9. Code of distribution plot

Besides the plot of density and distribution, I also conducted exploration about the relationship of income and other factors including age, hours-per-week and education-num and generated boxplot. Figure 10 shows the code of boxplot.Figure 11 shows the result of boxplot.

```
f, (ax1) = mp.subplots(figsize=(10, 10))
ax1 = sns.boxplot(x = 'income', y = 'age', data=clean_data1, showfliers=True)

f, (ax2) = mp.subplots(figsize=(10, 10))
ax2 = sns.boxplot(x = 'income', y = 'education-num', data=clean_data1, showfliers=True)

f, (ax3) = mp.subplots(figsize=(10, 10))
ax3 = sns.boxplot(x = 'income', y = 'hours-per-week', data=clean_data1, showfliers=True)
```

Figure 10. Code of box plot

According to the result of box plot, the average age of adults who have income of larger than 50K is higher than the adults who have income of less than 50K. The adults of higher education level are more likely to get income of larger than 50K according to the result of the boxplot. The situation is almost the same with hours-per-week which means the adults who have income larger than 50K also have longer working hours per week.

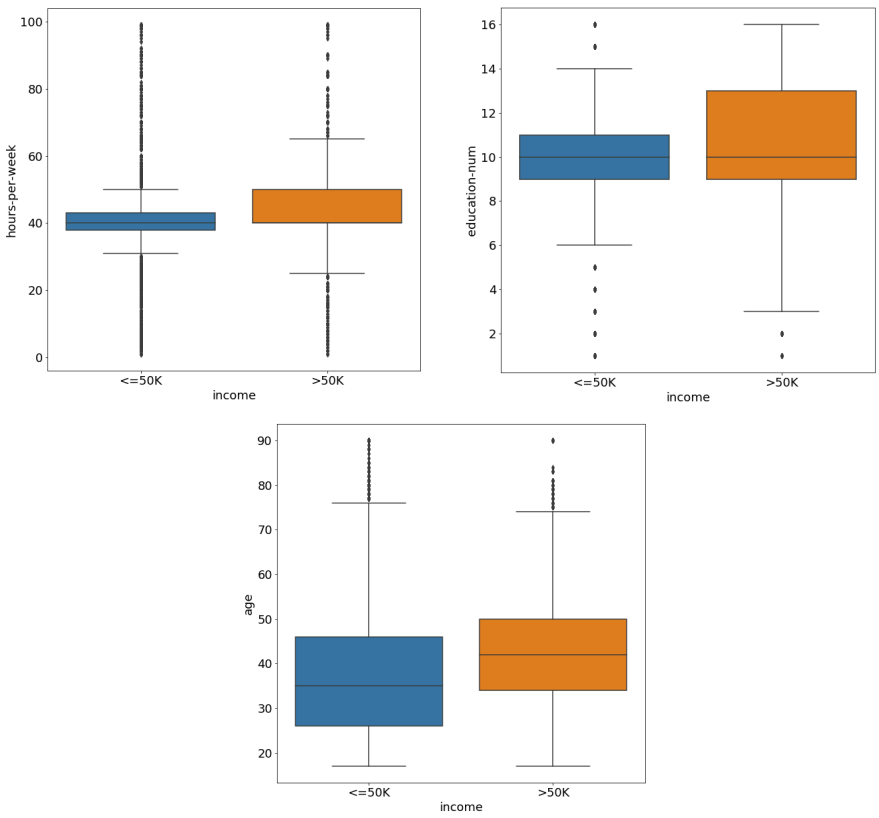


Figure 11. BoxPlot of relationship between income and factors

Verify the data quality

After basic exploration of dataset, it's also critical to ensure the quality of dataset. Before starting the modelling, what is more important it to ensure that the dataset does not contain errors like missing values, data errors and measurement errors. To avoid potential pitfalls, I conducted a quality analysis of available data throughly before modelling. Figure 14 shows the code which is used to conduct the testing.

- ***Missing data.*** Figure 12 displays the result of checking missing data. The attribute 'workclass','occupation' and 'native-country' has missing values with percentage of 6%, 6% and 2%.
- ***Data errors.*** I conducted the testing of checking if there is character like "?" In the dataset, and according to the testing of data errors, there is no such errors in the dataset.
- ***Measurement errors.*** In this project, zero values are taken as measurement errors. Figure 13 shows the result of checking zero values. There are 92 percent of records have zero values for attribute 'capital-gain' and 95 percent of that for attribute 'capital-loss'.

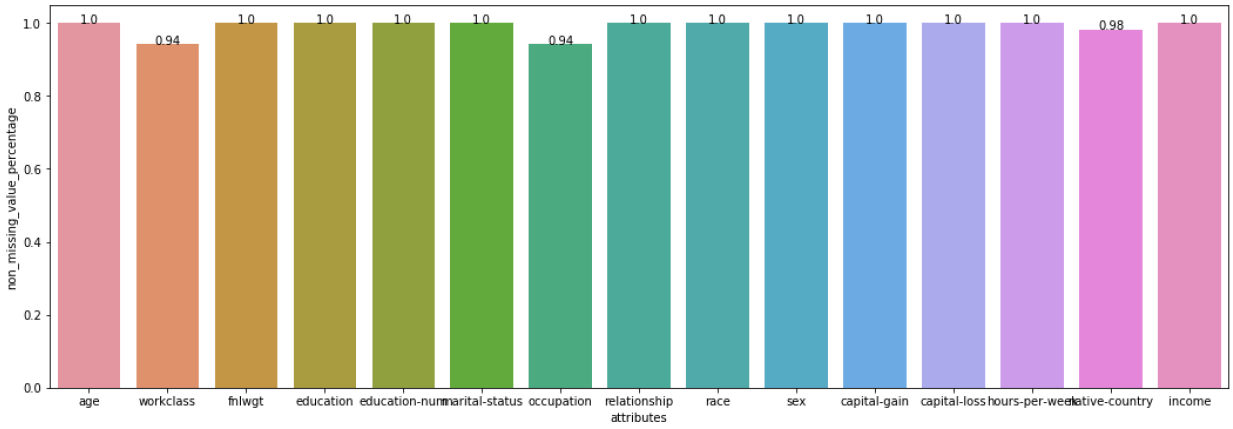


Figure 12. Percentage of missing data

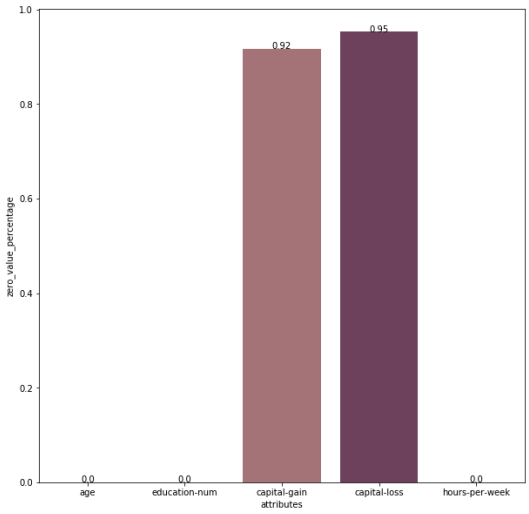


Figure 13. Percentage of measurement error

```
missing_values = frame.isna()

attr = []
percentage = []

for coll in missing_values.columns:
    print(coll)
    print(np.mean(missing_values[coll]))
    attr.append(coll)
    percentage.append(1-np.mean(missing_values[coll]))

result1 = pd.DataFrame(columns=['attributes','non_missing_value_percentage'])
result1['attributes'] = attr
result1['non_missing_value_percentage'] = percentage

f, (g) = mp.subplots(figsize=(18, 6))
g=sns.barplot(x='attributes',y='non_missing_value_percentage',data=result1)

for index, row in result1.iterrows():
    g.text(row.name,row.non_missing_value_percentage,
           round(row.non_missing_value_percentage,2), color='black', ha="center")

mp.show()
```

Figure 14. Code of data validation

Data preparation

Data preparation is one of the most important and often time-consuming aspects of data mining. In fact, this step takes about 60% of the time and effort of this project. According to the goals of the project, I conducted data preparation which involves several different tasks, including merging data, selecting a sample subset of data, removing blank or missing values and splitting the dataset into training and testing datasets.

Select the data

Selecting items. The study will be limited to the approximately 42882 adults, and I selected the records which the age of adult is between 18 to 60 and this will also reduce the data of outliers.

Selecting attributes. The basic line to select the attributes in this project is that the percentage of zero values can not be greater than 50 percent. According to the result of data quality, attributes like 'capital-gain' and 'capital-loss' have above 90 percent of zero values, so these two attributes will be dropped during the following steps. In the first iteration, all the attributes will be kept to see the result. Also, I conducted the testing of importance for features using different ways and results will be compared in the other iterations for this project. Details of this will be illustrated in Chapter 8.5.

Clean the data

According to the result of data validation of quality, there are missing data and measurement errors in the dataset, and the invalid data need to be cleaned before modelling.

Missing data. The attribute 'workclass','occupation' and 'native-country' has missing values with percentage of 6%, 6% and 2%.Data with missing values have been deleted from the dataset.

Measurement errors. There are 92 percent of records have zero values for attribute 'capital-gain' and 95 percent of that for attribute 'capital-loss'. So the two columns are dropped from datset.

Figure 15 shows the code which is used to conduct data cleaning.

```
clean_data = frame.drop(columns=['capital-gain', 'capital-loss'])

clean_data1 = clean_data.dropna()

for x in clean_data.columns:
    mv = clean_data[x].isin(["?"]).sum()
    if mv > 0:
        clean_data = clean_data[clean_data[x] != '?']
```

Figure 15. Code of data cleaning

Construct the data

The dataset satisfies the requirement of analyse. So no further work need to be done for this step.

Integrate various data sources

In this project, the format of dataset is a csv file which includes all the data and attributes and is divided into multiple csv file. To import the dataset into the project, the code is implemented to browse all the csv

files under the directory and the read each csv file in a loop and then merge the data to a whole dataframe. The process includes appending the data and merging the multiple dataset together. Figure 16 shows the code which is used to integrate the data sources.

```
path = '.'
all_files = glob.glob(path + "/*.csv")
li = []

for filename in all_files:
    df = pd.read_csv(filename, index_col=None, header=0)
    li.append(df)

frame = pd.concat(li, axis=0, ignore_index=True)
```

Figure 16. Code of data cleaning

Format the data as required

As a last step before modelling, it is useful to check whether certain procedures require a specific arrangement or request to the information. For instance, it isn't unprecedented that a succession calculation requires the information to be pre-sorted before running the model. For the models I used in this project, no particular data format or order is required, however, there are several categorial variables in the dataset, to proceed with the modelling, I transformed the categorial variables to be numeric by label encoder.

Data transformation

Reduce the data

In the dataset, the column “fnlwgt” is considered useless in this project, so I dropped this column. Also feature importance of all the attributes are generated using altirigom of decision tree. Figure 17 shows the feature importance.

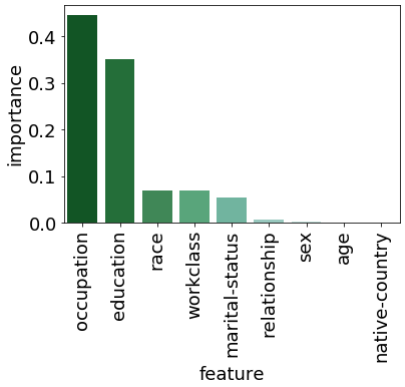


Figure 17. Feature Importance

The result shows that sex, age, and native-country has much less importance score than other attributes, they are supposed to be dropped. In the first iteration of this project, I kept all above attributes, and then dropped features of less importance to see the difference. Figure 18 shows the code which is used in this step.

```
model = se.RandomForestClassifier(n_estimators=50, max_depth=6,
                                  random_state=5)
model.fit(train_x, train_y)
print(model)
fi_dy = model.feature_importances_
print(fi_dy)
mp.figure('Random Forest')
mp.subplot(211)
mp.title('Random Forest', fontsize=16)
mp.ylabel('Importance', fontsize=12)
mp.tick_params(labelsize=10)
mp.grid(axis='y', linestyle=':')
sorted_indices = fi_dy.argsort()[::-1]
pos = np.arange(sorted_indices.size)
mp.bar(pos, fi_dy[sorted_indices])
mp.xticks(pos, processed_data.columns[sorted_indices],
          rotation=90)
mp.tight_layout()
mp.show()
```

Figure 18. Code of data cleaning

Project the data

After selecting the importance feature, new dataset is generated. Besides transforming the categorical variables in to labels, there is no variable which need statistical transformation. The new created dataset is now able to be used in modelling.

Data-mining method selection

Modelling is generally directed in numerous cycles. Commonly, information diggers run a few models using the default parameters and afterward tweak the parameters or return to the information readiness stage for controls required by the model of decision. It is uncommon for a specialist's information mining question to be addressed acceptably with a solitary model and a solitary execution. In this section, I firstly matched the data mining methods and then selected the data mining methods.

Match data mining methods

Supervised, unsupervised and reinforcement are three main methods of machine learning. Supervised and unsupervised are mainly used to deal with projects of machine learning. Reinforcement learning is more powerful and complex to apply for problems.Figure 19 shows the structure of methods of matching learning.(Madhu Sanjeevi, 2017)

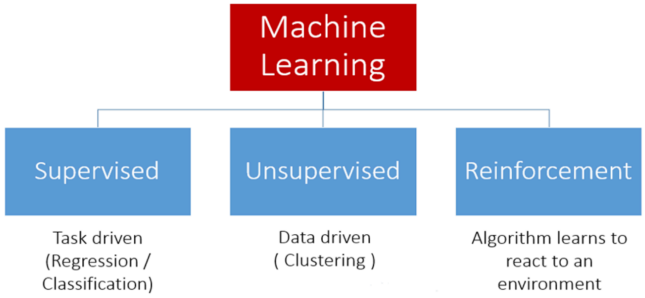


Figure 18. Methods of Matching Learning

Supervised learning is the machine learning task of learning a capacity that maps a contribution to a yield dependent on model information yield pairs. It gathers a capacity from labeled training data consisting of a set of training examples. In directed adapting, every model is a pair consisting of an information object (regularly a vector) and an ideal yield esteem (likewise called the supervisory signal). A regulated learning calculation breaks down the preparation information and produces an induced capacity, which can be used for mapping new models. An ideal situation will consider the calculation to accurately decide the class marks for concealed cases. This requires the taking in calculation to sum up from the preparation information to concealed circumstances in a "sensible" manner. Figure 19 shows the main types of supervised learning.(Madhu Sanjeevi, 2017)

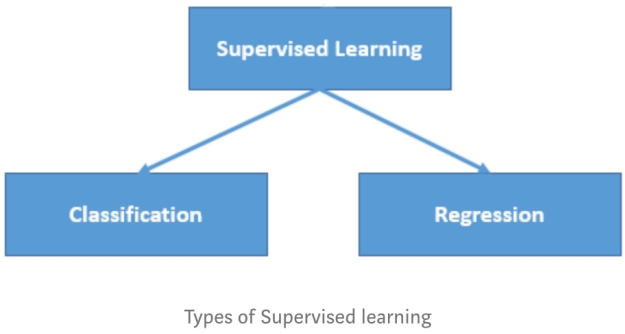


Figure 19. Types of Supervised learning

Regression: This is a type of problem where we need to predict the *continuous-response* value

Classification: This is a type of problem where we predict the *categorical response* value where the data can be separated into specific "**classes**". Classification and regression trees are machine-learning techniques which are used to develop forecast models from information. The calculation recursively parcel the information space and fit a basic expectation model inside each segment. A chart of choice tree will be produced by the calculation. Classification trees and regression trees are two fundamental segments of the calculation. A limited number of unordered qualities, with forecast mistake estimated as far as misclassification cost are taken as contributions of arrangement trees. Consistent or requested discrete qualities, with forecast mistake commonly estimated by the squared distinction between the watched and anticipated qualities are taken as contributions of relapse trees.(Loh, 2011).

Classification, which is one of the most important learning models in data mining, aims at building a model to predict future behaviours through classifying datasets into multiple predefined classes based on certain criteria. Neural networks, decision trees and if-then-else rules are common tools which are used for classification (Ngai, 2009).

Unsupervised learning

Unsupervised learning may be a sort of machine learning algorithm utilized to draw deductions from datasets comprising of information data without checked responses. The foremost broadly recognized unaided learning strategy is cluster examination, which is utilized for exploratory data examination to discover concealed illustrations or gathering in data. The bunches are illustrated employing an extent of resemblance which is characterized upon estimations, for case, Euclidean or probabilistic partition. The preparing information does not include Targets here so we don't tell the framework where to go , the framework should get it itself from the information we allow. Figure 20 appears the method of unsupervised learning. Figure 19 shows the process of unsupervised learning. (Madhu Sanjeevi, 2017)

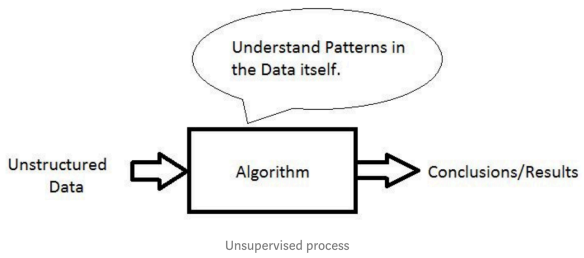


Figure 19.Unsupervised process

Select data-mining method

Based on the above discussion about the architecture of methods for machine learning, I made decisions about which ones to use. The most appropriate mode is determined based on the following considerations:

- ***The data types available for mining.*** The data types of input for this project are numeric and categorical and the categorical variables will be transformed to labels before the modelling.The data type of target is categorical variable.
- ***Data mining goals.*** There are two main steps for the objectives of this project. One is to create new model to train the dataset and distinguish it into different groups. After this, the model is used to predict if the income of an adult will be higher than 50K. The objectives match the method of classification among different data mining methods.
- ***Specific modelling requirements.*** No particular size of data is required for the model but it does require variables to be taken as target. Easily presentable results are needed after building the model.

According to above discussion, classification is the appropriate method of data mining for this project.

Data-mining algorithm selection

Conduct exploratory analysis and discuss

After selecting the data-mining method, I conducted several different algorithms of classification including Logistic Regression, KNN, SVM, Decision Tree, Random Forests, Naive Bayes, Linear Discriminant Analysis and analysed the result of all the algorithms to find the best algorithm for the project.

Logistic Regression

Logistic regression is the proper relapse examination to direct when the reliant variable is dichotomous (binary). It is used when the dependent target is categorical. Like all regression investigations, logistic regression is a prescient analysis. It is utilized to depict information and to clarify the connection between one ward twofold factor and at least one ostensible, ordinal, interim or proportion level autonomous factors. In logistic regression, a solitary result variable Yi (I = 1,..., n) pursues a Bernoulli likelihood function that takes on the worth 1 with likelihood πi and 0 with likelihood $1 - \pi i$. At that point πi changes

over the perceptions as a backwards strategic capacity of a vector xi , which incorporates a steady and k − 1 illustrative variables.(Statistical Solution, 2019)

The objective of logistic regression is to locate the best fitting (yet organically sensible) model to depict the connection between the dichotomous normal for intrigue (subordinate variable = reaction or result variable) and a lot of autonomous (indicator or informative) factors. Strategic relapse produces the coefficients (and its standard blunders and importance levels) of an equation to anticipate a logit transformation of the likelihood of essence of the normal for intrigue. (Medcalc, 2019)

Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values. Figure 20 showed the following code to create a model of logistic regression. (Medcalc, 2019)

```
In [10]: logmodel = LogisticRegression()
         logmodel.fit(train_x,train_y)
         print(logmodel)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

Figure 20. Create logistic regression model

KNN

The examination of KNN technique has been turning into a hot research theme in information mining and AI since the calculation was proposed in 1967. To apply for the conventional kNN technique in enormous information, the past written works can be frequently ordered into two sections, i.e., quick finding the closest examples and choosing agents tests (or evacuating a few examples) to lessen the figuring of kNN. For example, Zhang proposed a Certainly Factor (CF) measure to manage the unsatisfactory quality of slanted class circulation in kNN techniques. Li et al. proposed a densitybased technique for diminishing the measure of preparing information. Zhao et al. proposed another calculation dependent on the utilization of marked examples and include the screening procedure condition, it causing the new calculation in time intricacy to have essentially diminished, and no huge impact on calculation result. These strategies were for the most part applied for quick search, measurement decrease, and improving

the effectiveness of the calculations. kNN calculation registers the separation between each preparation test and test tests in the dataset and after that profits k nearest tests. Its time unpredictability is straightly and is ensured to discover accurate k closest neighbors. In any case, the computational intricacy of the straight search technique is relative to the size of the preparation dataset for each test, it is O(nd), where n is the size of the preparation dataset and d is the dimensionality. This intricacy is costly for huge information. Since the kNN technique isn't preparing process, we propose to present another preparation procedure for kNN, which squares preparing dataset by a grouping calculation with straight multifaceted nature. During the testing procedure, for each test, we discover the k closest bunch focuses and direct a grouping for each test, and after that build another order model base on each group. In especially, the examples inside a bunch has high likeness. In this way, contrasting with the conventional kNN strategy, the proposed calculation decreases the time unpredictability of KNN, yet additionally doesn't include essentially impact characterization exactness. Figure 21 shows the code which is used to create a model of KNN.(Z. Deng, 2016)

```
#KNN
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(train_x, train_y)
print(neigh)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Figure 21. Create KNN model

SVM

The SVM classifier is a run of the mill discriminative classifier. Unique in relation to generative classifier, it for the most part centers around how well they can isolate the positives from the negatives, and doesn't attempt to comprehend the fundamental data of the individual classes. The SVM classifier maps first the case x in a preparation set into a high dimensional space through a capacity Φ, at that point processes a choice capacity of the structure f(x) = + b by amplifying the separation between the arrangement of focuses Φ(x) to the hyperplane or set of hyperplanes parameterized by (w, b) while being steady on the preparation set. SVM is a hyperplane which can be utilized to isolate the positive information from the negative information with most extreme edge in the element space. The edge represents the separation

between the hyperplane with the closest of the positive and negative information. (Hwanjo Yu, 2002). SVMs expand the edge (Winston phrasing: the 'road') around the isolating hyperplane. The choice capacity is completely indicated by a (normally exceptionally little) subset of preparing tests, the help vectors.This turns into a Quadratic programming issue that is anything but difficult to comprehend by standard strategies (M. E. Mavroforakis, 2006). Figure 22 shows the ideas of SVM.

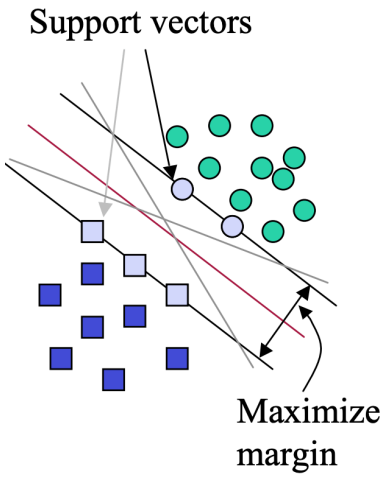


Figure 22. SVM Idea

A SVM finds the best isolating (maximal edge) hyperplane between the two classes of preparing tests in the element space, as it is appeared in Fig. 1. A direct discriminant capacity has the type of the straight practical , which relates to a hyperplane, separating the component space. On the off chance that, for a given example mapped in the component space to , the estimation of is a positive number, at that point the example has a place with the class named by the numeric worth ; else, it has a place with the class with worth.(Hwanjo Yu, 2002). Figure 23 shows the code is used to create a model of SVM.

```
#SVM
clf = svm.SVC(gamma='scale',probability=True)
clf.fit(train_x, train_y)
print(clf)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

Figure 23. Create SVM model

Decision Tree

Decision trees are among the notable AI methods. A decision tree is made out of three essential components. A decision node which is indicating a test characteristic. An edge or a branch which is relating to the one of the conceivable trait esteems which means one of the test quality results. A leaf which is additionally named an answer hub, contains the class to which the item has a place (N. Amor, 2004). Figure 24 shows the idea of decision tree.

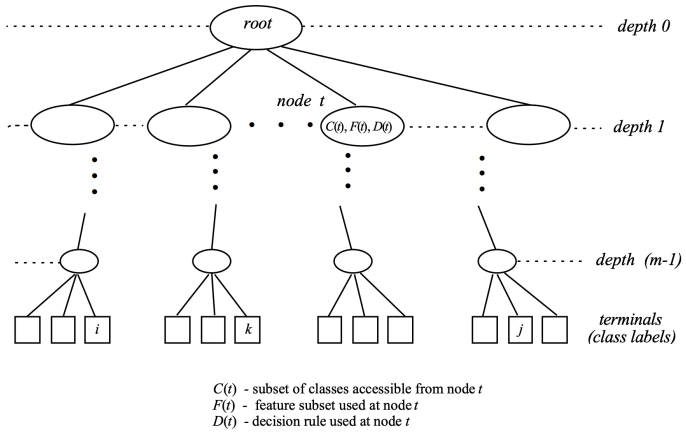


Figure 24. Decision Tree algrithom

In decision trees, two noteworthy stages ought to be guaranteed: 1. Building the tree. In view of a given preparing set, a choice tree is fabricated. It comprises of choosing for every choice hub the 'proper' test credit and furthermore to characterize the class marking each leaf. 2. Grouping. So as to group another example, we start by the foundation of the choice tree, at that point we test the characteristic determined by this hub. The aftereffect of this test permits to descend the tree limb with respect to the characteristic estimation of the given occurrence. This procedure will be rehashed until a leaf is experienced. The occasion is then being ordered in a similar class as the one describing the arrived at leaf (Shi, H, 2011). Figure 25 shows the code which is used to create a model of decision tree.

```
                        #DecisionTree
                        dtc = DecisionTreeClassifier()
                        dtc.fit(train_x,train_y)
                        print(dtc)
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

Figure 25. Create model of decision tree

Random Forests

There are three main steps for the algorithm of random forest. 1. Draw ntree bootstrap tests from the first information. 2. For every one of the bootstrap tests, grow an unpruned arrangement or relapse tree, with the accompanying change: at every hub, as opposed to picking the best split among all indicators, arbitrarily test of the indicators and pick the best split from among those factors. (Stowing can be thought of as the uncommon instance of arbitrary backwoods acquired, the quantity of indicators.) 3. Anticipate new information by conglomerating the expectations of the ntree trees (i.e., dominant part votes in favor of characterization, normal for relapse). A gauge of the blunder rate can be gotten, in light of the preparation information, by the accompanying: 1. At each bootstrap emphasis, anticipate the information not in the bootstrap test (what Breiman calls "out-of-pack", or OOB, information) utilizing the tree developed with the bootstrap test. 2. Total the OOB expectations. (On the normal, every datum bring up be out-of-pack around 36% of the occasions, so total these expectations.) Calcuate the blunder rate, and consider it the OOB gauge of mistake rate. The randomForest bundle alternatively delivers two extra snippets of data: a proportion of the significance of the indicator factors, and a proportion of the inside structure of the information (the vicinity of various information focuses to each other). Variable significance This is a troublesome idea to characterize by and large, on the grounds that the significance of a variable might be because of its (perhaps mind boggling) association with different factors. The arbitrary timberland calculation assesses the significance of a variable by taking a gander at how much expectation mistake increments when (OOB) information for that variable is permuted while all others are left unaltered. The vital computations are done tree by tree as the irregular timberland is built. The instinct is that "comparative" perceptions ought to be in a similar terminal hubs more frequently than unique ones. (Liaw A, 2002). Figure 26 shows the code which is used to create a model of random forests:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=2, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

```
#RandomForest
model = se.RandomForestClassifier(n_estimators=100, max_depth=2,
                                  random_state=0)
model.fit(train_x, train_y)
print(model)
```

Figure 26. Create model of random forest

Naive Bayes

Bayes frameworks are one of the most by and large used graphical models to address and deal with questionable information. Bayes frameworks are dictated by two portions: - A graphical part made out of an organized non-cyclic diagram (DAG) where vertices address events and edges are relations between events. - A numerical part including in an assessment of different associations in the DAG by a prohibitive probability allocation of each center with respect to its people. Honest Bayes are astoundingly clear Bayes frameworks which are made out of DAGs with only one root center (called parent), addressing the secretly center, and a couple of youths, contrasting with watched centers, with the strong assumption of self-governance among child center points concerning their parent. The portrayal is ensured by accepting the parent center to be a disguised variable communicating to which class each article in the testing set should have a spot and adolescent center points address different qualities showing this thing. In this manner, in closeness of a readiness set we should simply process the unforeseen probabilities since the structure is unique. At the point when the framework is estimated, it is possible to describe any new thing giving its characteristics using the Baye's standard. The Naïve Bayes request technique is gotten in this examination essentially for its ability to manage missing features, which occurs for a part of the neuropsychological assessments. A Naïve Bayes classifier is a fundamental probabilistic classifier reliant on the utilization of Bayes' speculation (portrayed numerically underneath) with the assumption of probabilistic self-rule between each pair of features; essentially this is rarely legitimate, as explicit features can be connected, yet Naïve Bayes classifiers display astoundingly

generous execution on features which are not cautiously self-sufficient (N. Amor, 2004). Figure 27 shows the code which is used to create Gaussian model.

```
#Gaussian Classifier
model_G = GaussianNB()
model_G.fit(train_x,train_y)
print(model_G)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Figure 27. Create model of Bayes

Linear Discriminant Analysis

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two commonly used techniques for data classification and dimensionality reduction. Linear Discriminant Analysis easily handles the case where the within-class frequencies are unequal and their performances has been examined on randomly generated test data. This method maximizes the ratio of between-class variance to the within-class variance in any particular data set thereby guaranteeing maximal separability. The use of Linear Discriminant Analysis for data classification is applied to classification problem in speech recognition.We decided to implement an algorithm for LDA in hopes of providing better classification compared to Principal Components Analysis. The prime difference between LDA and PCA is that PCA does more of feature classification and LDA does data classification. In PCA, the shape and location of the original data sets changes when transformed to a different space whereas LDA doesn't change the location but only tries to provide more class separability and draw a decision region between the given classes.This method also helps to better understand the distribution of the feature data (Balakrishnama, S, 2001). Figure 28 shows the code which is used to create the LDA model.

```
#LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(train_x,train_y)
print(lda)
```

```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage=None,
                           solver='svd', store_covariance=False, tol=0.0001)
```

Figure 28. Create model of LDA

MLP

Multi-Layer Perceptrons (MLP) are completely associated feedforward nets with at least one layers of hubs between the info and the yield hubs. Each layer is made out of at least one counterfeit neurons in parallel. A neuron, has N weighted sources of info and a solitary yield. A neuron consolidates these weighted contributions by shaping their entirety and, with reference to a limit worth and actuation work, it will decide its yield. Figure 29 shows the idea of this algorithm.

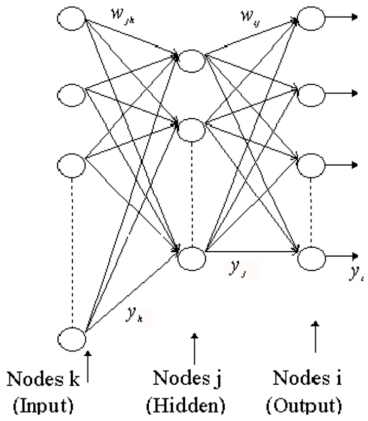


Figure 29. MLP algorithm

The net is prepared by at first choosing little arbitrary loads and inside limits, and showing all preparation information more than once. Loads are balanced after each preliminary utilizing data determining the right class until loads unite and the cost capacity is diminished to a worthy worth. The by and large great execution found for the back proliferation calculation is to some degree astounding thinking about that it is a slope drop system that may locate a neighbourhood least in the cost capacity rather than the ideal worldwide least (Riedmiller, 1994). This algorithm will be used in one of the multiple iterations in Chapter 8.

Select data-mining algorithms

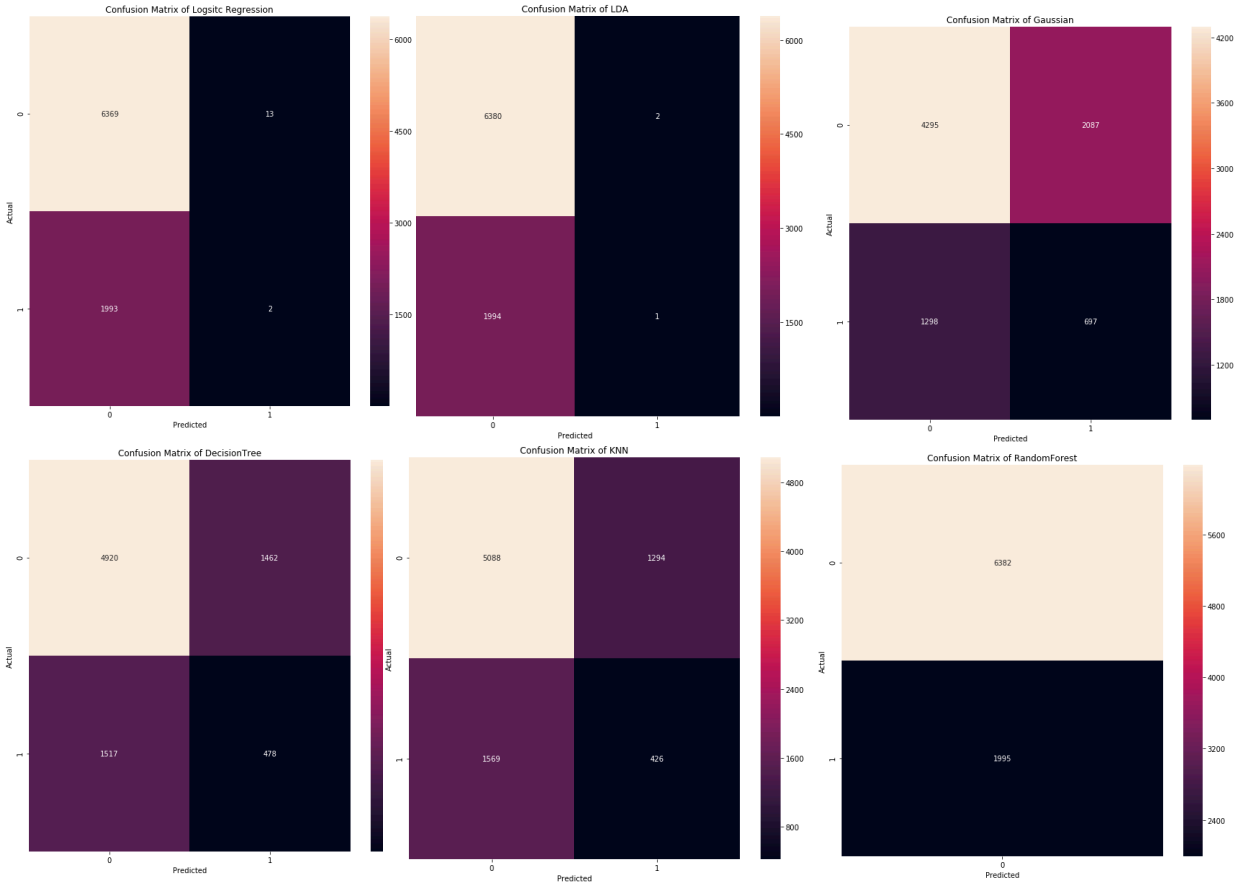
To select the data-mining algorithms, I executed all the models, and evaluated the models with different ways including Classification Report, Logarithmis Loss and Confusion Matrix. The details of results will be illustrated in this section.

Confusion Matrix

A confusion matrix is an outline of expectation results on an order issue. The quantity of right and erroneous forecasts are condensed with check esteems and separated by each class. This is the way in to the disarray matrix.The perplexity framework demonstrates the manners by which your order model is befuddled when it makes expectations. It gives us understanding not just into the mistakes being made by a classifier however more significantly the kinds of blunders that are being made. The following code is used to generate the confusion matrix of all the models and plot with heat map (O. Caelen, 2017). Figure 30 shows the code which is used to plot the confusion matrix. Figure 31 shows the result of the code.

```
for pre_y in pred_y_list:
    df = pd.DataFrame(test_y,pre_y)
    confusion_matrix = pd.crosstab(test_y,pre_y, rownames=['Actual'],
                                   colnames=['Predicted'])
    #print(confusion_matrix)
    mp.figure(figsize=(10,10))
    mp.title('Confusion Matrix of '+ alghrithm_list[count])
    sns.heatmap(confusion_matrix, annot=True, fmt="d")
    count += 1
```

Figure 30. MLP algorithm



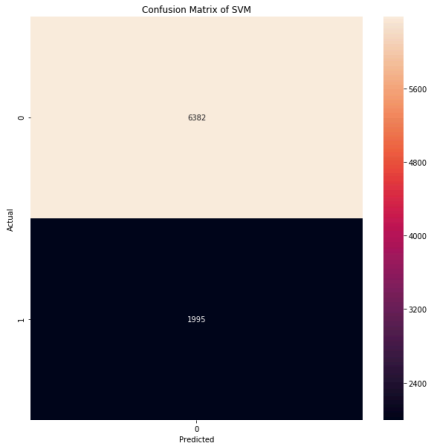


Figure 31. Result of confusion matrix

According to the result of confusion matrix, not easy to find the model which is relatively better, so I conducted classification report and logarithmic loss to get better result.

Classification Report

The classification report visualizer displays the precision, recall, F1, and support scores for the model. To support easier interpretation and problem detection, the report integrates numerical scores with a color-coded heatmap. All heatmaps are in the range (0.0, 1.0) to facilitate easy comparison of classification models across different classification reports (Yellowbrick, 2019).

Precision is the ability of a classiifer not to label an instance positive that is actually negative. For each class it is defined as as the ratio of true positives to the sum of true and false positives. Said another way, "for all instances classified positive, what percent was correct?" (Yellowbrick, 2019).

Recall is the ability of a classifier to find all positive instances. For each class it is defined as the ratio of true positives to the sum of true positives and false negatives. Said another way, "for all instances that were actually positive, what percent was classified correctly?" (Yellowbrick, 2019).

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. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy (Yellowbrick, 2019).

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. Support doesn't change between models but instead diagnoses the evaluation process. Figure 32 shows the code which is used to get the result of classification report of all the models. Figure 33 shows the result of classification report (Yellowbrick, 2019).

```
pred_y_list = [pred_test_y,neigh_pred_test_y,svm_pred_test_y,
               gaussian_predicted_test_y,dtc_y_pred,lda_y_pred,log_y_pred]

alghrithm_list = ["RandomForest", "KNN", "SVM", "Gaussian", "DecisionTree",
                  "LDA", "Logsitc Regression"]
count = 0

for pred_y in pred_y_list:
    print("The classification report of " + alghrithm_list[count])
    print(classification_report(test_y, pred_y, target_names=target_names))
    count += 1
```

Figure 32. Code of classification report

```
The classification report of SVM
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86      6382
     class 1       0.00      0.00      0.00      1995

    accuracy                           0.76      8377
   macro avg       0.38      0.50      0.43      8377
weighted avg       0.58      0.76      0.66      8377

The classification report of Gaussian
              precision    recall  f1-score   support

     class 0       0.77      0.67      0.72      6382
     class 1       0.25      0.35      0.29      1995

    accuracy                           0.60      8377
   macro avg       0.51      0.51      0.50      8377
weighted avg       0.64      0.60      0.62      8377

The classification report of DecisionTree
              precision    recall  f1-score   support

     class 0       0.76      0.77      0.77      6382
     class 1       0.25      0.24      0.25      1995

    accuracy                           0.64      8377
   macro avg       0.51      0.51      0.51      8377
weighted avg       0.64      0.64      0.64      8377

The classification report of LDA
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86      6382
     class 1       0.33      0.00      0.00      1995

    accuracy                           0.76      8377
   macro avg       0.55      0.50      0.43      8377
weighted avg       0.66      0.76      0.66      8377

The classification report of RandomForest
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86      6382
     class 1       0.00      0.00      0.00      1995

    accuracy                           0.76      8377
   macro avg       0.38      0.50      0.43      8377
weighted avg       0.58      0.76      0.66      8377

The classification report of KNN
              precision    recall  f1-score   support

     class 0       0.76      0.80      0.78      6382
     class 1       0.25      0.21      0.23      1995

    accuracy                           0.66      8377
   macro avg       0.51      0.51      0.50      8377
weighted avg       0.64      0.66      0.65      8377

The classification report of Logsitc Regression
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86      6382
     class 1       0.13      0.00      0.00      1995

    accuracy                           0.76      8377
   macro avg       0.45      0.50      0.43      8377
weighted avg       0.61      0.76      0.66      8377
```

Figure 33. Result of classification report

According to the result of classification report, there are several models including Random Forest, SVM, LDA and Logistic Regression, of which the f1-score is relatively higher than others, and the values are about 76%.

Logarithmic Loss

Logarithmic loss estimates the presentation of a grouping model where the expectation info is a likelihood esteem somewhere in the range of 0 and 1. The objective of our AI models is to limit this worth. An ideal model would have a log loss of 0. Log misfortune increments as the anticipated likelihood wanders from the genuine mark. So anticipating a likelihood of .012 when the real perception name is 1 would be terrible and bring about a high log misfortune(Wiki, 2019). There is a progressively point by point clarification of the avocations and math behind log misfortune.Figure 34 shows the code which is used to calculate the logarithmic loss of all the models and the result of it.

```
model_list = [model, neigh, clf, model_G, dtc, lda, logmodel]
alghrithm_list = ["RandomForest", "KNN", "SVM", "Gaussian", "DecisionTree",
                  "LDA", "Logsitc Regression"]
i = 0
for ml in model_list:
    probs = ml.predict_proba(test_x)[:, 1]
    score = log_loss(test_y, probs)
    print("The Logarithmic Loss of " + alghrithm_list[i] + "is : " + str(score))
    i += 1
```

```
The Logarithmic Loss of RandomForest is : 0.5900843779792563
The Logarithmic Loss of KNN is : 5.872219217694295
The Logarithmic Loss of SVM is : 0.558511905845494
The Logarithmic Loss of Gaussian is : 0.836993037032529
The Logarithmic Loss of DecisionTree is : 11.015286571610057
The Logarithmic Loss of LDA is : 0.5849516863603016
The Logarithmic Loss of Logsitc Regression is : 0.5921187534800798
```

Figure 34. Code and result of logarithmic loss

According to the result of calculation of logarithmic loss, there are several models including Random Forest, SVM, LDA, Gaussian and Logistic Regression, of which the value of Logarithmic Loss is relatively lower than others.

Select appropriate model and choose relevant parameter

According to the above analyse, I decided to proceed with several models with better performance which include Random Forest, SVM and Logistic Regression. Grid search is used to look for the best parameters

of all the models. Figure 35 shows the code is used to get the best parameters of Random Forest model and the result is also displayed by the following figure.

```
rfc=se.RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(train_x, train_y)
print(CV_rfc.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'n_estimators': 100}
```

Figure 35. Grid Search for Random Forest

For the limit of hardware, I conducted the testing of SVM model by trying out different parameters. Figure 36 shows the code is used to test SVM model and the result is also displayed by the following figure.

```
clf_f = svm.SVC(probability=True,kernel='linear',gamma=3,
                decision_function_shape='ovo',shrinking=True)
clf_f.fit(train_x, train_y)
pred_y_clf_f = clf_f.predict(test_x)
print(classification_report(test_y, pred_y_clf_f, target_names=target_names))

clf_f1 = svm.SVC(probability=True,kernel='poly',gamma=4,
                decision_function_shape='ovo',shrinking=True)
clf_f1.fit(train_x, train_y)
pred_y_clf_f1 = clf_f1.predict(test_x)
print(classification_report(test_y, pred_y_clf_f1, target_names=target_names))
```

```
                  precision    recall  f1-score   support

         class 0       0.76      1.00      0.86      6382
         class 1       0.00      0.00      0.00      1995

        accuracy                           0.76      8377
       macro avg       0.38      0.50      0.43      8377
    weighted avg       0.58      0.76      0.66      8377

    //anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning:

    Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.

                  precision    recall  f1-score   support

         class 0       0.76      1.00      0.86      6382
         class 1       0.00      0.00      0.00      1995

        accuracy                           0.76      8377
       macro avg       0.38      0.50      0.43      8377
    weighted avg       0.58      0.76      0.66      8377
```

Figure 36. SVM Model

Figure 37 shows the code which is used to get the best parameters of Logistic Regression model and the result is displayed by the following figure.

```
from sklearn.model_selection import GridSearchCV
lr = LogisticRegression()
grid_values = {'penalty': ['l1', 'l2'],'C':[0.001,.009,0.01,.09,1,5,10,25]}
grid_lr_acc = GridSearchCV(lr, param_grid = grid_values,scoring = 'recall')
grid_lr_acc.fit(train_x, train_y)
print(grid_lr_acc.best_params_)
```

```
{'C': 1, 'penalty': 'l1'}
```

Figure 37. Grid Search for Logistic Regression

Data Mining

Create and justify test designs

To finish the test, firstly, the dataset is imported and divided into two partitions which includes the training dataset and testing dataset. In the first iteration, the training dataset includes 70 percent of the total dataset and the left 30 percent of the total dataset is the testing dataset. This partition of dataset is taken as a basic testing, and different partition will be taken in the other iterations to get the best result. The target is income and all the other attributes are taken as input. Missing values are imputed and the outliers and extreme values are removed from the dataset to get more accurate result. After the analyse of different models, Random Forest, SVM and Logistic Regression are selected as the models to proceed with the following steps. Also, I will conduct data mining with the three chosen models and then try to evaluate and improve the results in different iterations.

Conduct data mining

According to the best parameters got from above steps, I conducted the data mining with three different models after finishing the partition of the dataset. Figure 38 shows the details of executing the models with the best parameters.

```
#Logsitc Regression
logmodel_n = LogisticRegression(C=1, penalty='l1')
logmodel_n.fit(train_x,train_y)
pred_y_lr = logmodel_n.predict(test_x)
print(classification_report(test_y, pred_y_lr, target_names=target_names))

clf_n = svm.SVC(gamma=3,probability=True,kernel='rbf',
                decision_function_shape='ovr',shrinking=True)
clf_n.fit(train_x, train_y)
pred_y_svm = clf_n.predict(test_x)
print(classification_report(test_y, pred_y_svm, target_names=target_names))

#RandomForest
model_rf = se.RandomForestClassifier(n_estimators=100, max_depth=6,
                                     random_state=42, max_features='auto')
model_rf.fit(train_x, train_y)
pred_y_rf = model_rf.predict(test_x)
print(classification_report(test_y, pred_y_rf, target_names=target_names))
```

Figure 38. Grid Search for Logistic Regression

Search for patterns

To get insight of the data, and find more patterns, figure 39 shows the code which is used to generate the correlation matrix. According to the matrix, there are some attributes which have correlation with the income attribute which is the target of the model. The attributes like work-class, education, maritial-status, occupation, relationship, race, sex, native-country are all related to the income of adults.

```
mp.figure(figsize=(10,10))
sns.heatmap(processed_data.corr(), annot=True,fmt=".2f")
```

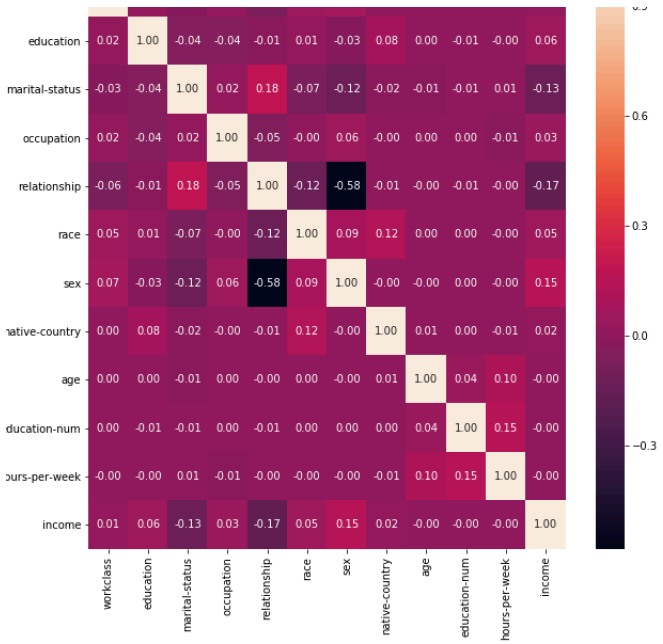


Figure 39. Correlation Matrix

Interpretation

Study and discuss the mined patterns

According to the result of correlation matrix, attributes like work-class, education, occupation, race, sex have affect on the value of income. This is almost the same with the result of feature importance got from models. It's also the foundation for further research about conducting more iterations to improve the model and find patterns like the relationship between income to these factors.

Visualise the data, results, models, and patterns

To analyse the mined pattern, more graphs of relations between multiple factors are generated to help on interpretation. Figure 40 shows the code which is used to create the heat map of the relations.

```
df1 = clean_data1[['marital-status','education-num','income']]

df2 = clean_data1[['workclass','education-num','income']]

df3 = clean_data1[['occupation','education-num','income']]


heatmap1_data = pd.pivot_table(df1, values='education-num',
                     index=['income'],
                     columns='marital-status')


heatmap2_data = pd.pivot_table(df2, values='education-num',
                     index=['income'],
                     columns='workclass')

heatmap3_data = pd.pivot_table(df3, values='education-num',
                     index=['income'],
                     columns='occupation')

mp.figure(figsize=(6,6))
sns.heatmap(heatmap1_data, cmap="YlGnBu")

mp.figure(figsize=(6,6))
sns.heatmap(heatmap2_data, cmap="RdBu")
```

Figure 40. Code of heat map

Figure 41 shows the relationship between income, marital-status and education-num.

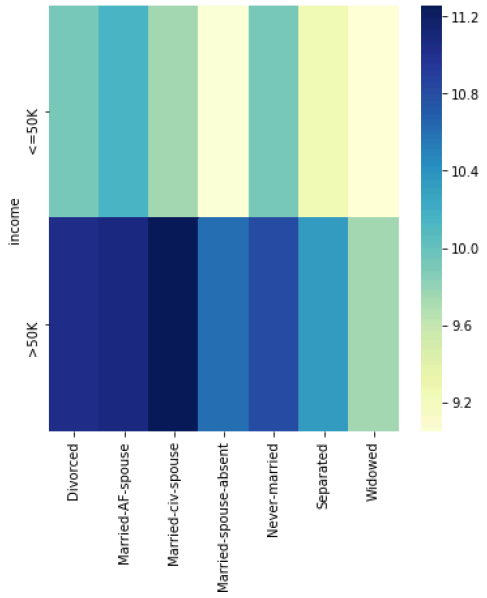


Figure 41. Heat Map of Income, Marital-status, Education

Figure 42 shows the relationship between income, education-num and workclass.

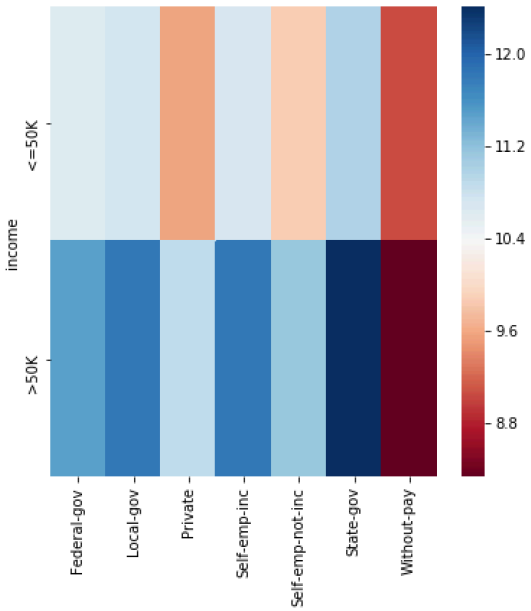


Figure 42. Heat Map of Income, Education-num, Workclass

Figure 43 displays the pattern between income, occupation and education-num.

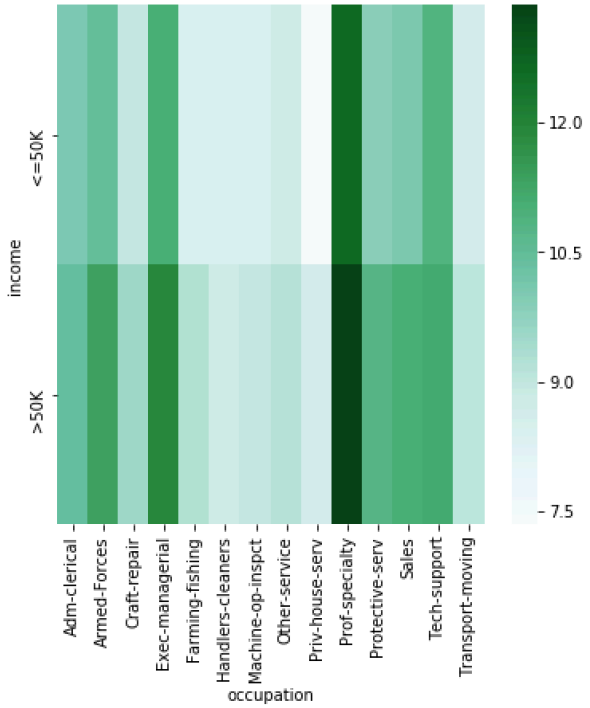


Figure 43. Heat Map of Income, Education-num, Workclass

Interpret the results, models, and patterns

According to the result of analyse, people with higher education level and married would have more chance to get income which is larger than 50K. At the same time, people who work for government with higher level of education would also have high salary. Age is not an important factor to income because people with different age would have various level of income.

Assess and evaluate results, models, and patterns

The model which has the highest f1-score is logistic regression in the experiment of the first iteration and the f1-score 0.76. Also more experiments and analyse will be conducted in the other iterations. Figure 44 shows the result of experiment.

```
              precision    recall  f1-score   support

     class 0       0.76      0.99      0.86      9604
     class 1       0.19      0.00      0.01      2962

    accuracy                           0.76     12566
   macro avg       0.47      0.50      0.44     12566
weighted avg       0.63      0.76      0.66     12566

              precision    recall  f1-score   support

     class 0       0.76      0.92      0.83      9604
     class 1       0.22      0.08      0.11      2962

    accuracy                           0.72     12566
   macro avg       0.49      0.50      0.47     12566
weighted avg       0.64      0.72      0.66     12566

              precision    recall  f1-score   support

     class 0       0.77      0.83      0.80      9604
     class 1       0.24      0.17      0.20      2962

    accuracy                           0.68     12566
   macro avg       0.50      0.50      0.50     12566
weighted avg       0.64      0.68      0.66     12566
```

Figure 44. Classification Report

Iterate prior steps

Based on the result after model assessment, it's time to have another look at the models. In this project, I conducted two steps:

- ***Adjust the parameters of existing models.***

During the step of building models, I proceeded the grid search to look for the best parameters for model of Logistic Regression and Random Forest, in this step, I changed the parameters of SVM to get different result and also changed the partition of dataset and selection of features , then dropped the features which are of lower importance like ‘age’,’native-country’.

Iteration 1

In this step, I changed the partition of the dataset by setting the parameters of partition to be 0.5 to look into the difference of the result. No much difference for the result of Logistic Regression and the f1 score is around 76%. For SVM and RandomForest, the f1 score is slightly different but still around 70%. Figure 45 shows the code of changing the partition and Figure 46 shows the result.

```
su.shuffle(x, y, random_state=7)
train_size = int(len(x) * 0.5)
train_x, test_x, train_y, test_y = x[:train_size], x[train_size:], y[:train_size], y[train_size:]
target_names = ['class 0', 'class 1']
```

Figure 45.Code of setting partition

```
Classification Report of Logstic Regression :
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86     15866
     class 1       0.00      0.00      0.00      5076

    accuracy                           0.76     20942
   macro avg       0.38      0.50      0.43     20942
weighted avg       0.57      0.76      0.65     20942

Classification Report of SVM :
              precision    recall  f1-score   support

     class 0       0.77      0.90      0.83     15866
     class 1       0.36      0.17      0.23      5076

    accuracy                           0.72     20942
   macro avg       0.57      0.54      0.53     20942
weighted avg       0.67      0.72      0.69     20942

Classification Report of RandomForest :
              precision    recall  f1-score   support

     class 0       0.78      0.89      0.83     15866
     class 1       0.39      0.21      0.28      5076

    accuracy                           0.73     20942
   macro avg       0.59      0.55      0.56     20942
weighted avg       0.69      0.73      0.70     20942
```

Figure 46. Result of Classification Report

Iteration 2

In this step, I changed the partition of the dataset by setting the parameters of partition to be 0.6 to look into the difference of the result. No much difference for the result of Logistic Regression and the f1 score is around 76%. For SVM and RandomForest, the f1 score is slightly different but still around 70%. Figure 47 shows the code of changing the partition and Figure 48 shows the result.

```
su.shuffle(x, y, random_state=7)
train_size = int(len(x) * 0.6)
train_x, test_x, train_y, test_y = x[:train_size], x[train_size:], y[:train_size], y[train_size:]
target_names = ['class 0', 'class 1']
```

Figure 47.Code of setting partition

```
Classification Report of Logstic Regression :
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86     12736
     class 1       0.00      0.00      0.00      4018

    accuracy                           0.76     16754
   macro avg       0.38      0.50      0.43     16754
weighted avg       0.58      0.76      0.66     16754

Classification Report of SVM :
              precision    recall  f1-score   support

     class 0       0.77      0.89      0.82     12736
     class 1       0.30      0.15      0.20      4018

    accuracy                           0.71     16754
   macro avg       0.53      0.52      0.51     16754
weighted avg       0.66      0.71      0.67     16754

Classification Report of RandomForest :
              precision    recall  f1-score   support

     class 0       0.77      0.90      0.83     12736
     class 1       0.31      0.14      0.20      4018

    accuracy                           0.72     16754
   macro avg       0.54      0.52      0.51     16754
weighted avg       0.66      0.72      0.68     16754
```

Figure 48.Code of setting partition

Iteration 3

In this step, I changed the parameter of SVM classifier and the f1 score was improved to be around 76%. The result also indicated that, for this project, SVM classifier with linear kernel and ovo decision function shape is fitting better to the model. Figure 49 shows the code of changing the parameter of SVM and Figure 50 shows the result.

```
#Logsitc Regression
logmodel_n = LogisticRegression(C=1,penalty='l1')
logmodel_n.fit(train_x,train_y)
pred_y_lr = logmodel_n.predict(test_x)
print("Classification Report of Logstic Regression : ")
print(classification_report(test_y, pred_y_lr, target_names=target_names))

#SVM
clf_n = svm.SVC(gamma=5,probability=True,kernel='linear',
                decision_function_shape='ovo',shrinking=True)
clf_n.fit(train_x, train_y)
pred_y_svm = clf_n.predict(test_x)
print("Classification Report of SVM : ")
print(classification_report(test_y, pred_y_svm, target_names=target_names))

#RandomForest
model_rf = se.RandomForestClassifier(n_estimators=100, max_depth=6,
                                     random_state=42, max_features='auto')
model_rf.fit(train_x, train_y)
pred_y_rf = model_rf.predict(test_x)
print("Classification Report of RandomForest : ")
print(classification_report(test_y, pred_y_rf, target_names=target_names))
```

Figure 49. Code of setting parameters of SVM

```
Classification Report of Logstic Regression :
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86     12736
     class 1       0.00      0.00      0.00      4018

    accuracy                           0.76     16754
   macro avg       0.38      0.50      0.43     16754
weighted avg       0.58      0.76      0.66     16754


Classification Report of SVM :
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86     12736
     class 1       0.00      0.00      0.00      4018

    accuracy                           0.76     16754
   macro avg       0.38      0.50      0.43     16754
weighted avg       0.58      0.76      0.66     16754


Classification Report of RandomForest :
              precision    recall  f1-score   support

     class 0       0.77      0.90      0.83     12736
     class 1       0.31      0.14      0.20      4018

    accuracy                           0.72     16754
   macro avg       0.54      0.52      0.51     16754
weighted avg       0.66      0.72      0.68     16754
```

Figure 50. Result of Classification Report

- ***Choose a different model to address data mining problem.***

Besides changing the test plan of the project, I also tried to conduct the experiment with MLP model, the following code is used to create the model and the result is also displayed. The accuracy is also about 76% which means the model is fitting well but there is still work need to do to improve the model. Figure 51 shows the code of building MLP model and Figure 52 shows the result.

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.neural_network import MLPClassifier

mlp = MLPClassifier(solver='lbfgs', alpha=1e-5,
                    hidden_layer_sizes=(6, 3), random_state=10)

model = mlp.fit(train_x, train_y)

y_pred = model.predict(test_x)
print("Accuracy:",metrics.accuracy_score(test_y, y_pred))
print(classification_report(test_y, y_pred, target_names=target_names))
```

Figure 51. Code of building MLP model

```
Accuracy: 0.7618479169153635
              precision    recall  f1-score   support

     class 0       0.76      1.00      0.86      6382
     class 1       0.00      0.00      0.00      1995

    accuracy                           0.76      8377
   macro avg       0.38      0.50      0.43      8377
weighted avg       0.58      0.76      0.66      8377
```

Figure 52. Result of Classification Report

Conclusion

As per the above strides, there are a few factors which are of larger amount of significance inlacing model, relationship, occupation, marital-status, education. Individuals who have more elevated amount of training and work for government would have progressively opportunity to get pay which is bigger than 50K. Python is chosen to be used as programming suite to process the information and finish the investigate. Logistic Regression can be used to precede with this exploration and the f1 score of the model is about 0.76 which is higher than that of different models.

Further Work

To apply the knowledge and deploy the implementation, government should be notified about the conclusion from the project. Assuming that they accept the results of the study, education should be of higher priority for the contribution of cities.

To monitor and maintain the implementation, the attributes like relationship, occupation, marital-status should be tracked in the future. The code of the project was implemented in python packages which can be used to get the validity and accuracy of the model. A model of f1 score less than 0.7 will be considered as expired. After the model is expired, the suit of the code can be used to run the models and get the new model which has highest accuracy.

To enhance the solution in the future, more iterations can be used to execute the project automatically when the dataset is updated. More complex and strong visualisation can be applied to find the deeper patten between different factors.

References

Ngai, E. W. T., Xiu, L., & Chau, D. C. K. (2009). Application of data mining techniques in customer Relationship management: A literature review and classification. Expert Systems with Applications, 36(2, Part 2), 2592-2602.

Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang. PEBL: positive example based learning for web page classification using SVM. In KDD, pages 239–248. ACM, 2002.

Loh (2011), Classification and regression trees, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery.

C L Blake, C J Merz. UCI repository of machine learning databases University of California, Irvine, Department of Information and Computer Sciences. 1998

Jakovljevic, N., et al. "Comparison of linear discriminant analysis approaches in automatic speech recognition." *Elektronika ir Elektrotechnika*, vol. 19, no. 7, 2013, p. 76+. *Gale Academic Onefile*. Accessed 13 Sept. 2019.

Shi, H., and Liu, Y. (2011). Naïve Bayes vs. support vector machine: resilience to missing data, in Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7003 LNAI(PART 2) (Taiyuan), 680–687.

J. Wang, S. J. Redmond, M. Bertoux, J. R. Hodges, and M. Hornberger, “A comparison of magnetic resonance imaging and neuropsychological examination in the diagnostic distinction of Alzheimer’s disease and behavioral variant frontotemporal dementia,” Frontiers in Aging Neuroscience, vol. 8, article 119, 2016.

O. Caelen, “A Bayesian interpretation of the confusion matrix”, Annals of Mathematics and Artificial Intelligence, 81(3–4), 429–450, 2017.

Riedmiller, M.A. (1994). Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms.

Z. Deng, X. Zhu, D. Cheng, M. Zong, and S. Zhang, “Efficient KNN classification algorithm for big data,” Neurocomputing, vol. 195, pp. 143–148, Jun. 2016.

M. E. Mavroforakis and S. Theodoridis, “A geometric approach to Support Vector Machine (SVM) classification,” IEEE Trans. Neural Netw., vol. 17, no. 3, pp. 671–682, May 2006.

Safavian, S. R., and Landgrebe, D., 1991, A survey of decision tree classifier methodology. IEEE T ransactions on Systems, Man, and Cybernetics, 21, 660–674.

Liaw A, Wiener M (2002). “Classification and Regression by randomForest.” R News, 2(3), 18–22.

N. Amor, S. Benferhat, and Z. Elouedi. Naive bayes vs decision trees in intrusion detection systems. In Proceedings of the 2004 ACM symposium on Applied computing, pages 420–424. ACM, 2004.

Balakrishnama, S., Ganapathiraju “Linear Discriminant Analysis – A Brief Tutorial” Institute for Signal and Information Processing, 1998.

King, Gary, and Langche Zeng. 2001. “Logistic Regression in Rare Events Data.” Political Analysis, in press.

Madhu Sanjeevi. (2017). Different types of Machine learning and their types. Retrieved from www.medium.com/deep-math-machine-learning-ai/different-types-of-machine-learning-and-their-types-34760b9128a2

Statistical Solution. (2019). What is Logistic Regression. Retrieved from www.statisticssolutions.com/what-is-logistic-regression/

Medcalc. (2019). Logistic regression. Retrieved from www.medcalc.org/manual/logistic_regression.php

R. Berwick. (2019). An Idiot’s guide to Support vector machines (SVMs). Retrieved from web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf

Rajesh S. Brid . (2018). Introduction to Decision Trees. Retrieved from www.medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb

Yellowbrick. (2019). Classification Report. Retrieved from www.scikit-yb.org/en/latest/api/classifier/classification_report.html

GeeksforGeeks. (2019). Confusion Matrix in Machine Learning. Retrieved from www.geeksforgeeks.org/confusion-matrix-machine-learning/

Wiki. (2019). Log Loss. Retrieved from www.wiki.fast.ai/index.php/Log_Loss

I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html).

I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data.