# Introduction

In this assignment, I will be using machine learning with Python to create machine learning models to predict the survivability of a Titanic passenger and to recognise whether the given image. These two tasks will be separated into Part 1 and 2.

# Methodology

## Titanic passenger survivability machine learning model

### Getting the Data

First, we have to get the dataset of the titanic passengers which will be the data that will be fed to the machine learning model. The datasets are acquired from Kaggle. Kaggle holds a competition on who can create the best machine learning model to predict the survivability of the Titanic passengers, and they only provided the training and test datasets (train.csv and test.csv) for people to build a prediction machine learning model. A lot of different implementations of machine learning models are made for this application.

The dataset contains 11 features and 1 target variable (survived). The features and their meanings are:

* PassengerId - Passenger ID
* Pclass - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
* Name - Name (Have titles such as Mr, Mrs,
* Sex - Sex
* Age - Age
* SibSp - Number of Siblings/Spouses Aboard
* Parch - Number of Parents/Children Aboard
* Ticket - Ticket Number
* Fare - Passenger Fare
* Cabin - Cabin
* Embarked - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
* **Survived - Survival (0 = No; 1 = Yes)**

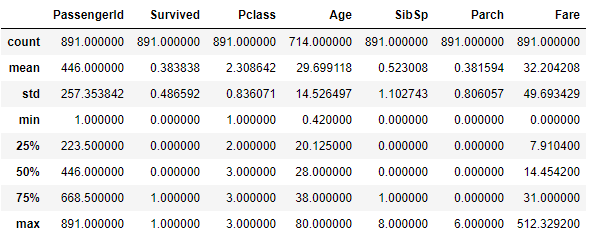
****

Figure: statistics of numerical features

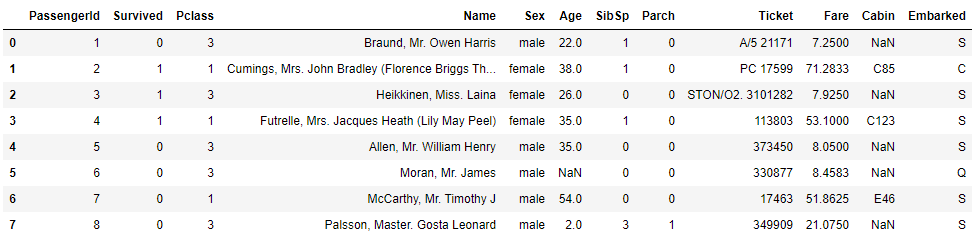
****

Figure: 8 examples of the raw data located inside the train.csv file

### Data Visualization

Before we start building our machine learning model, it is important to be able to visualize the data inside the csv to see the relationship between the features and the target variable. From analyzing each feature, we can also better filter out those features that are not useful in our machine learning model, and we are also able to determine whether there are any missing data in our dataset.

#### Age and Sex against Survived

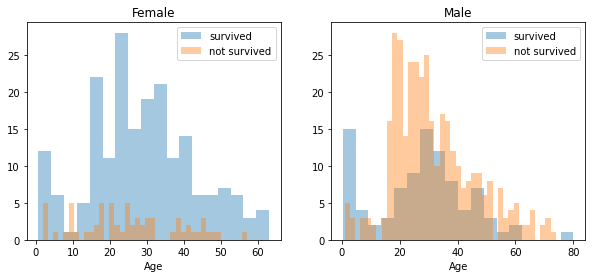


Figure : Histogram of Age and Sex against Survived

For male, most of them who survived are between 20 and 40 years old, however the most deaths are also between 16 to 40 years old. We can also see that a lot of the younger children survived from the Titanic crash. For women, more women survived the Titanic crash compared to men. This means that if you are a woman, you have a much higher probability to survive the Titanic crash. This may be due to the evacuation strategy for the Titanic, which does not have enough lifeboats for all the passengers, thus the crew allowed the women and children first onto the lifeboat, which is a valid reason for the higher probability of survival for women and children.

#### Passenger class against Survived

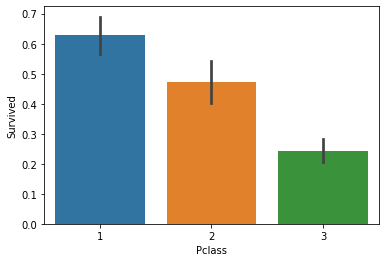


Figure: This bar graph shows the passenger class against the probability of survival

We can see here clearly that a passenger’s class contributes significantly to the survivability of a passenger. 1st Class has the highest probability, followed by 2ns class, then the last is 3rd class. This is the case because during the evacuation, the crew allowed 1st class passengers to board the lifeboats first, then only the 2nd class passengers. 3rd class passengers are last and by that time it is too late to board the lifeboats.

#### Siblings/ Spouse and Parents/ Children aboard against survived

For these 2 features, I have combined them into one feature called relatives, since these two features tell whether or not the passenger aboard the Titanic alone or with relatives. After, that I will plot the number of relatives against survivability to see if there is any meaningful correlation between these two variables.

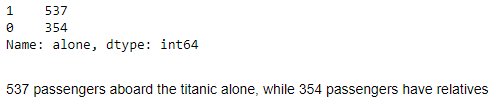


Figure: the number of passengers alone or not

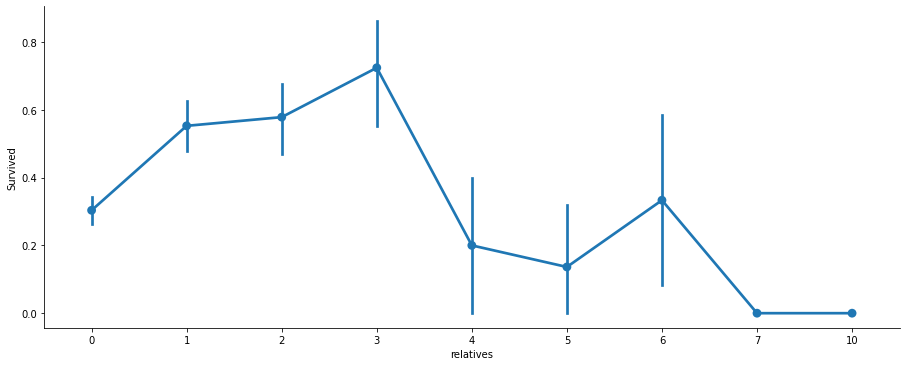


Figure: Line graph of the number of relatives against survivability

As we can see, the highest probability of surviving are passengers who have 1 to 3 siblings, with the rest having below 0.4 mean probability of surviving the Titanic crash.

### Data Preprocessing

The next step is to do data preprocessing before feeding it into our machine learning algorithm. The aim of data preprocessing is to prepare our data in the best possible way so that our machine learning model produces the best accuracy possible. Even some problems with the data can affect the accuracy of the machine learning algorithm greatly. The data preprocessing that we will do are:

* Solve missing data
* Drop features that won't help in predicting the survivability
* Change categorical features into integers using hot one encoding so that the machine learning model is able to understand
* Feature scaling since features have wide variety of ranges

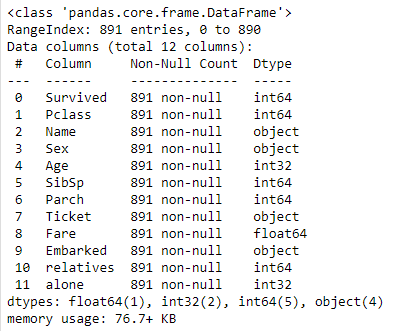


Figure: Features and the data types

#### Missing Data

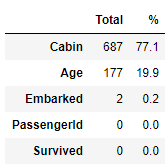


Figure : Features and the missing data

* Cabin has 687 missing data, which is 77.1% of the total data. Age has 177 missing data which accounts for 17%. Embarked only has 2 missing data. Certain strategies have to be used to fill in the missing values here.
* For “age”, I replaced the missing values with values in the range between **mean + standard deviation** and **mean - standard deviation.**
* For “embarked”, I replaced the missing values with the most common embarked location, which is Southampton (S).
* For “cabin” there are too many missing data so we will remove this feature since filling them may add inaccuracies to our machine learning algorithm.

#### Not so useful data

* Dropped PassengerID
* Dropped tickets
  + Everyone’s ticket is unique so it does not provide any meaningful correlation to the target variable.
* Dropped Name
  + However, in the “name” feature it includes titles such as
    - Mr
    - Miss
    - Mrs
    - Master
    - Rare
  + These titles may contribute to our machine learning algorithm

#### Feature conversion

* Convert “Fare” from float to int
* Convert “sex” from object to int
* Convert “Embarked” from object to int

Feature Scaling

* If different features have a huge difference in the range of their values, it will affect greatly to certain algorithms
  + Algorithms based on Gradient Descent such as linear regression, logistic regression, neural network and others
  + Algorithms based on Distance Between Data Points such as KNN, K-means and SVM
  + Tree- Based Algorithms
* There are a few ways to do feature scaling, such as normalization, standardization, or even group certain ranges of a feature into a new feature - e.g. Age can be grouped into Age Groups.
* Age and Fare have a wide range of values.
  + Range:
    - Age: 0.42 - 80 years old
    - Fare: 0 - 512.3292 pounds
  + To solve this huge variance in range of values, we will group Age and Fare into two new features respectively, Age Group and Fare Range.
  + The age will be grouped into 8 Age Groups of:
    - 0 < Age <= 11
    - 11 < Age <= 18
    - 18 < Age <= 22
    - 22 < Age <= 27
    - 27 < Age <= 33
    - 33 < Age <= 40
    - 40 < Age <= 66
    - 66 < Age
  + The Fare will be grouped into 4 Fare Ranges based on pandas qcut function.
    - 0 < Fare <= 7.896
    - 7.896 < Fare <= 14.454
    - 14.454 < Fare <= 31.5
    - 31.5 < Fare <= 512.329
  + After that, we will convert the Fare Range from float to int to simplify.

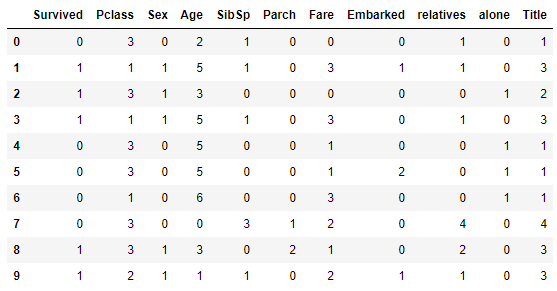


Figure: The dataset after feature conversion

### Building Machine Learning Models

After undergoing feature conversion, the data is ready to be fit into different machine learning algorithms. We will be fitting the data to 8 different machine learning algorithms:

* Stochastic Gradient Descent (SGD)
* Random Forest
* Logistic Regression
* K Nearest Neighbor
* Gaussian Naive Bayes
* Perceptron
* Linear Support Vector Machine
* Decision Tree

We will fit the training data to the machine learning algorithm. Then the machine learning model will try to predict the target variable given the features in the test dataset. The accuracy of each algorithm is shown below.

#### Results acquired

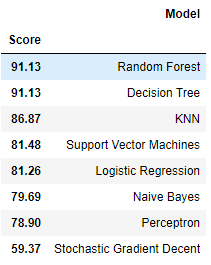


Figure: Accuracy achieved

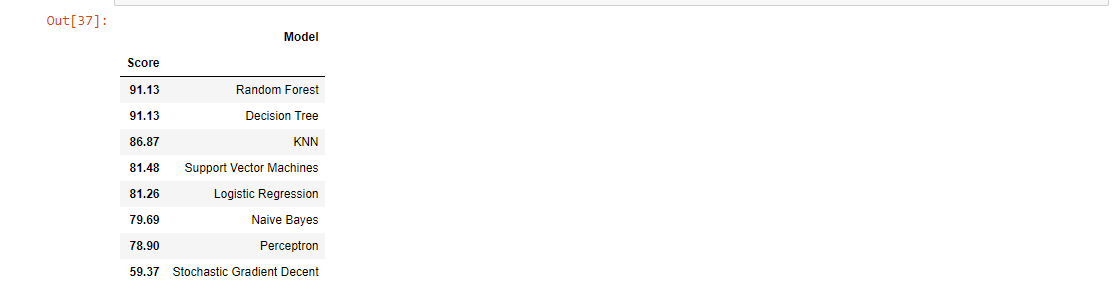
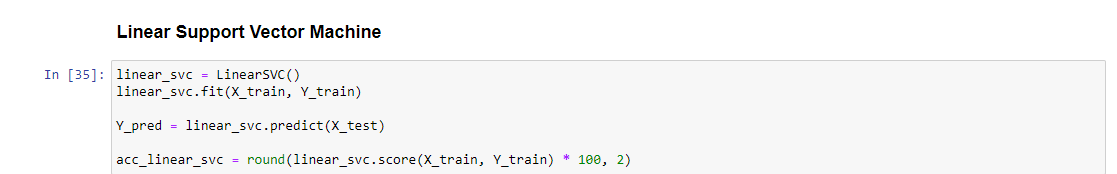
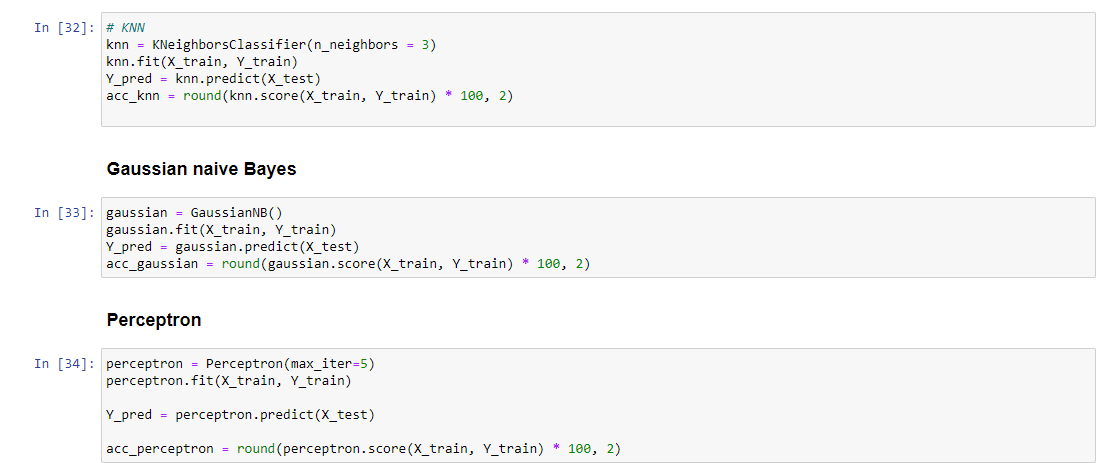
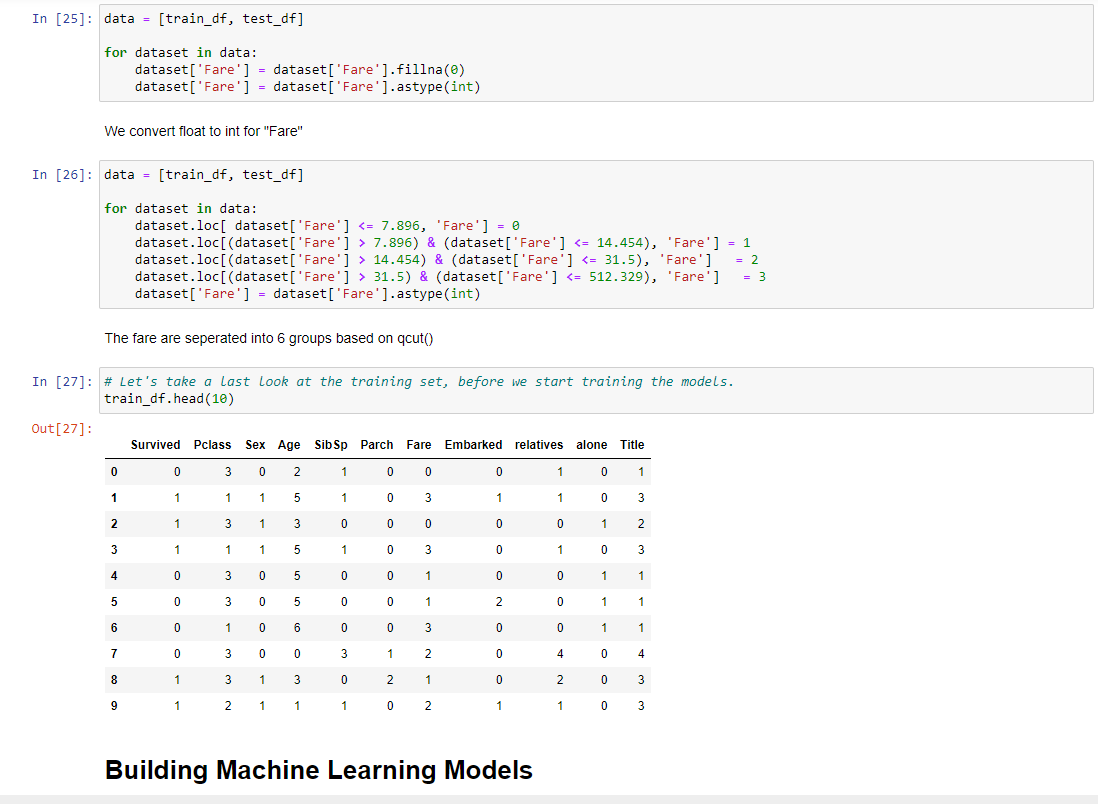
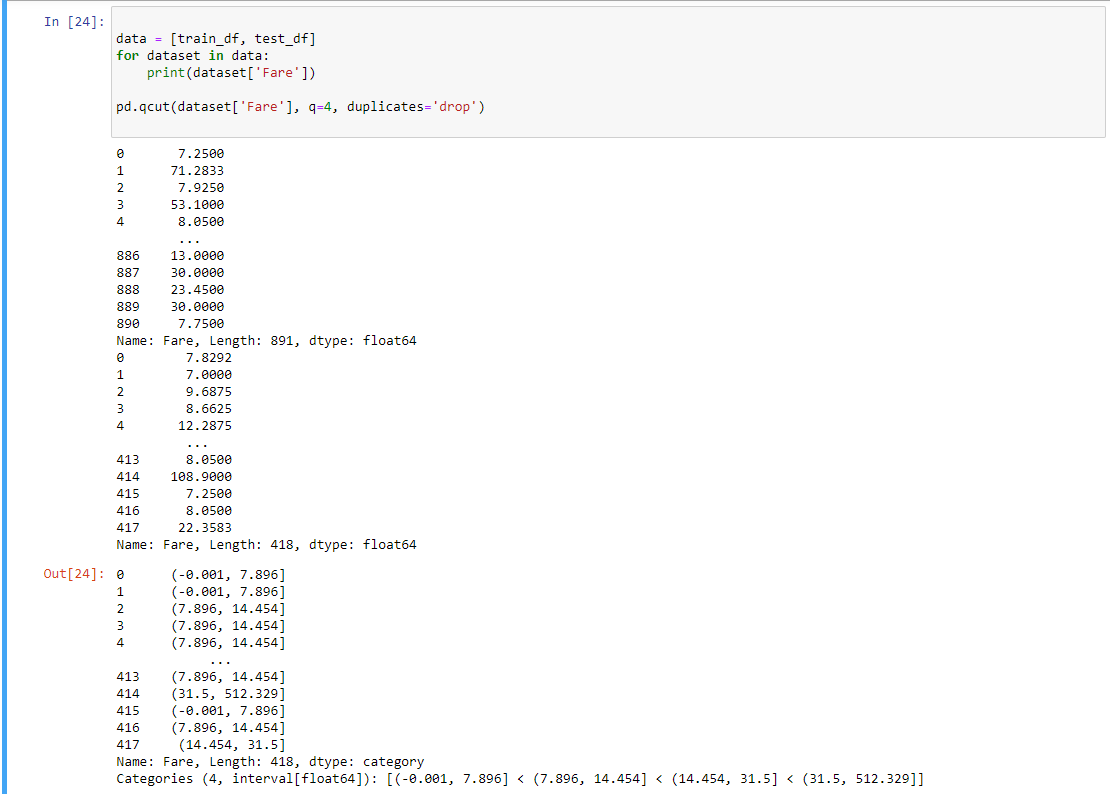
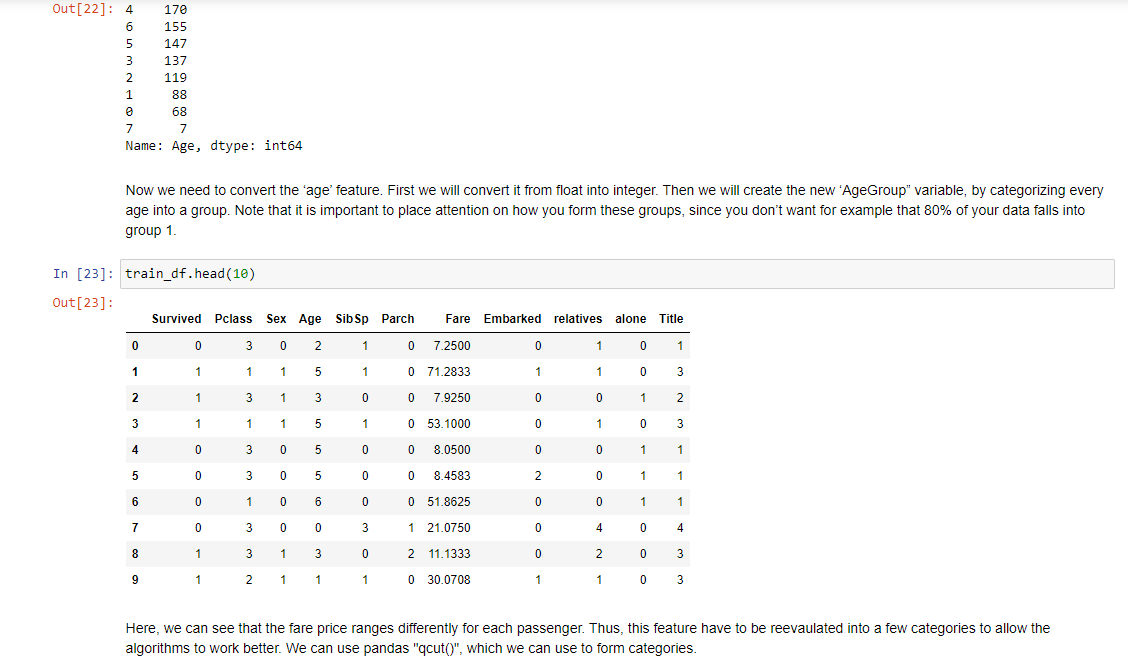
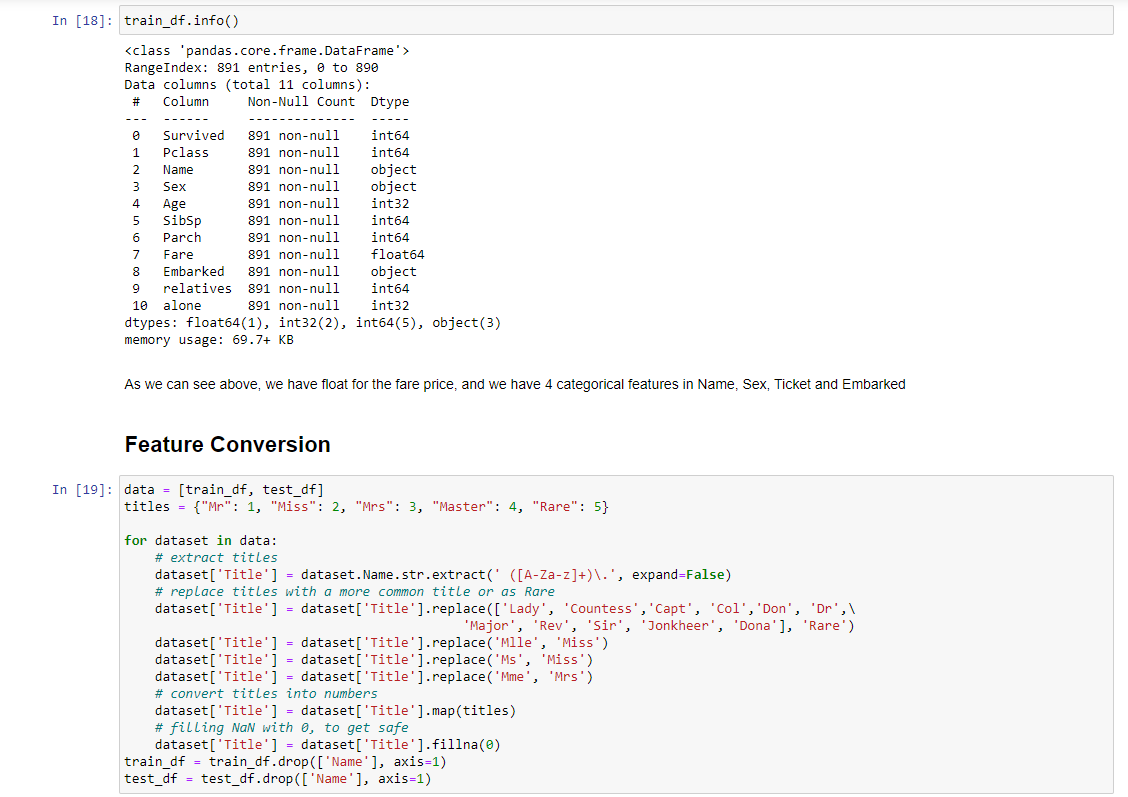
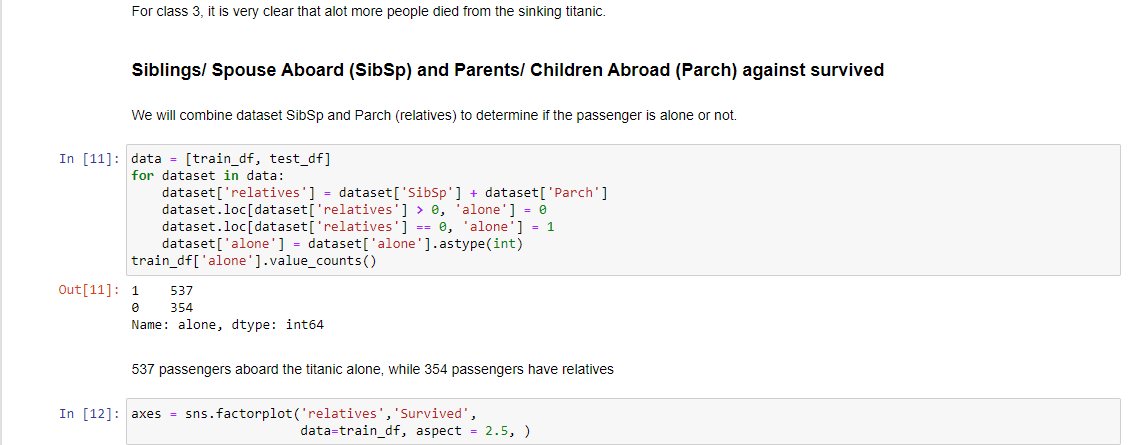
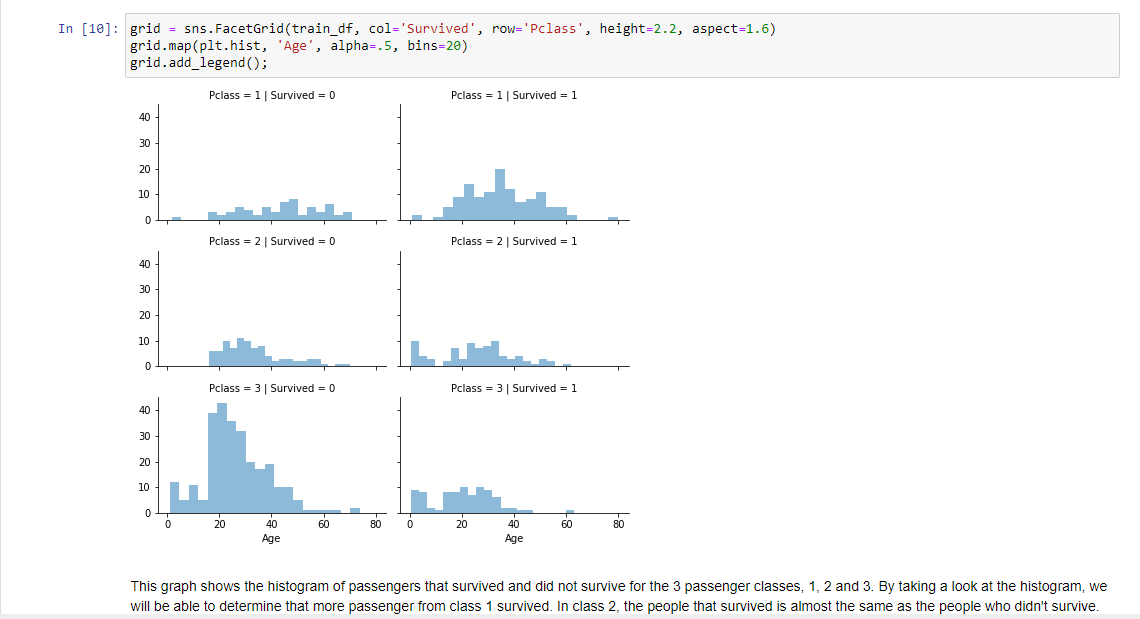
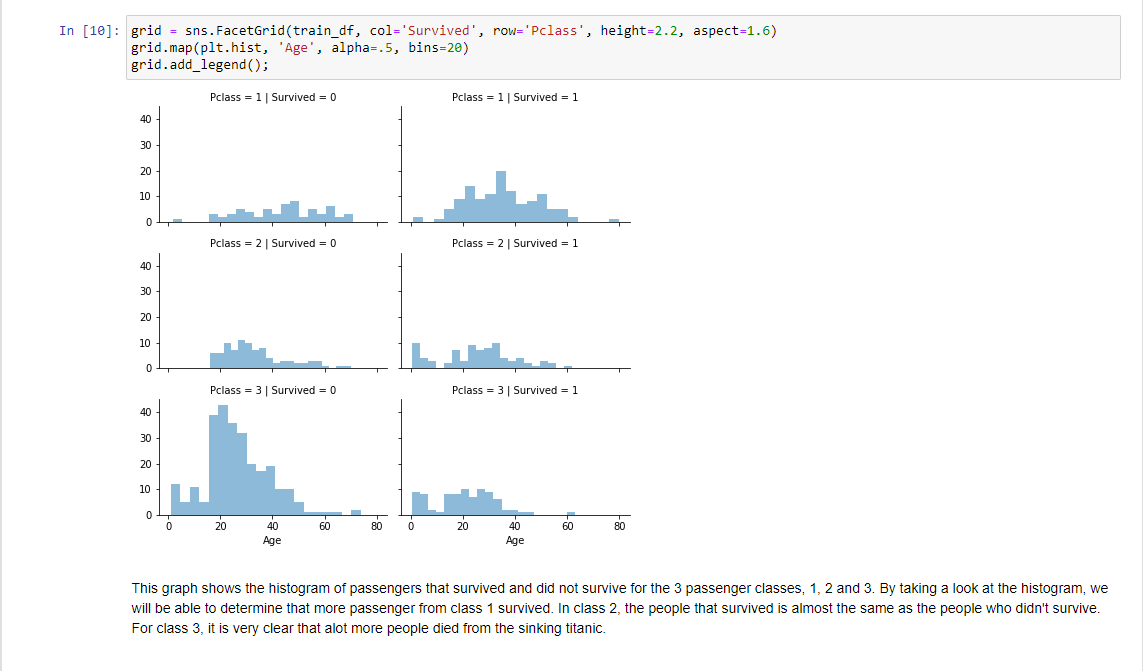
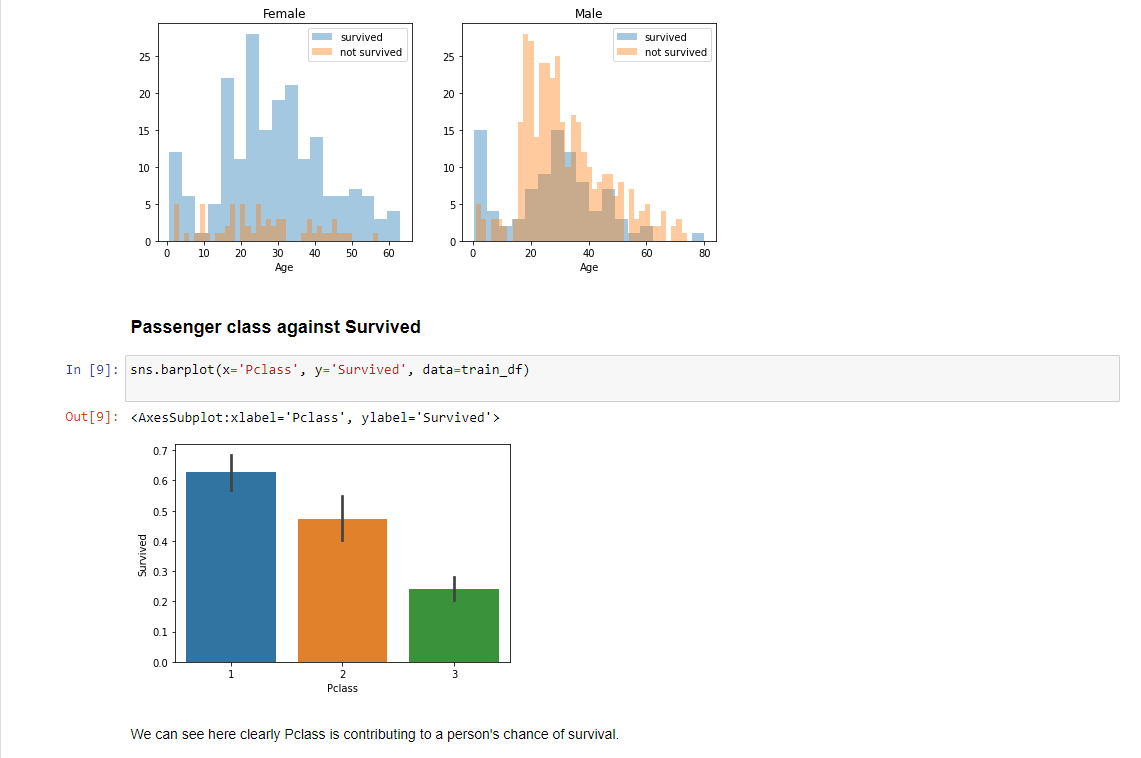
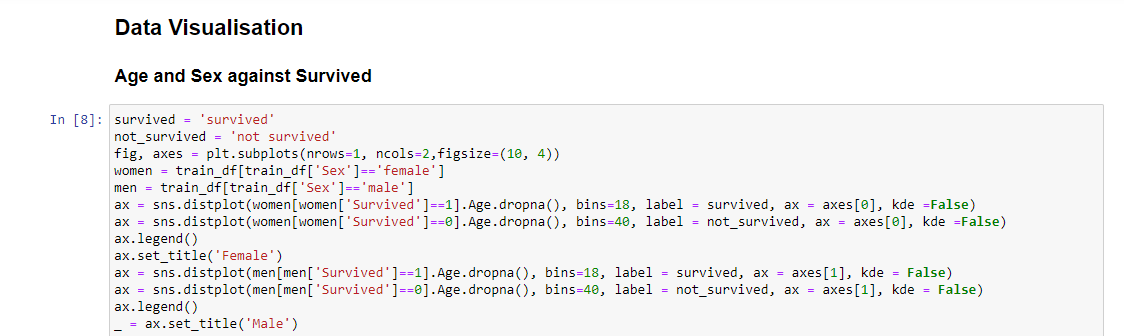
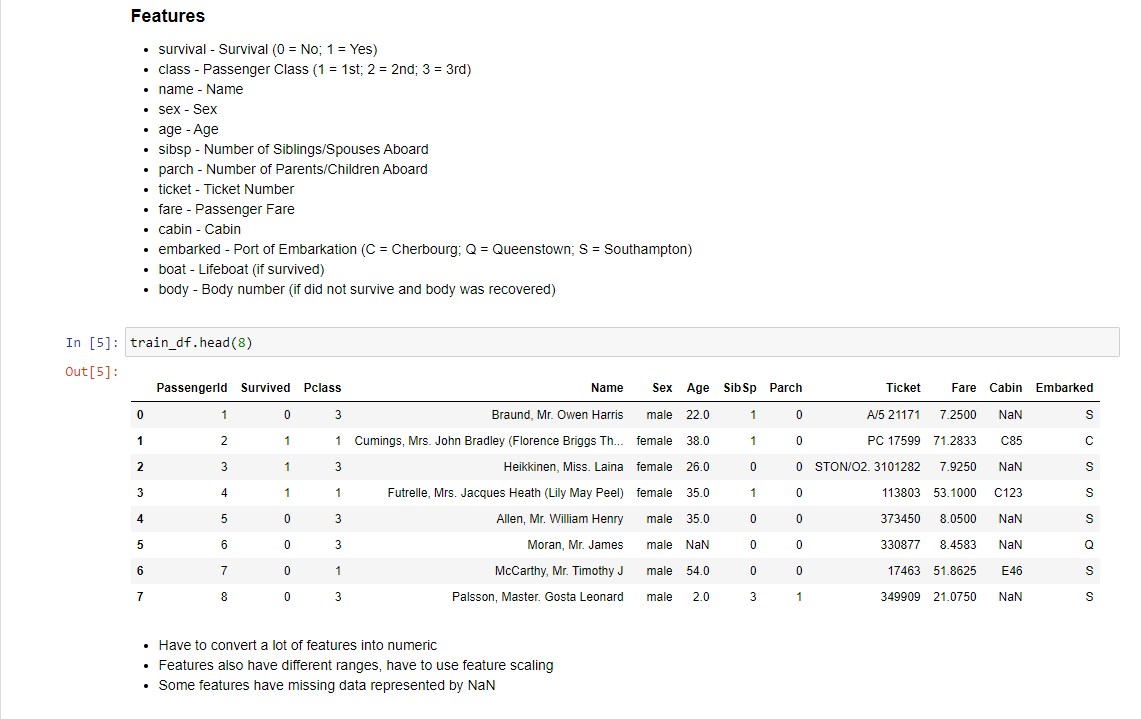
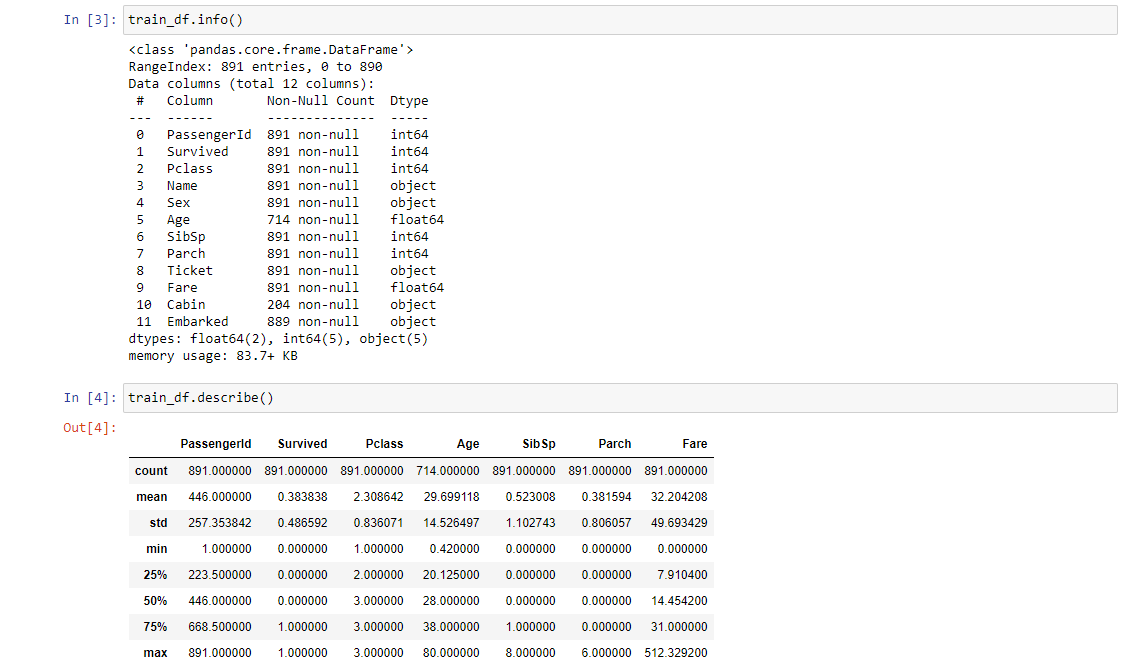
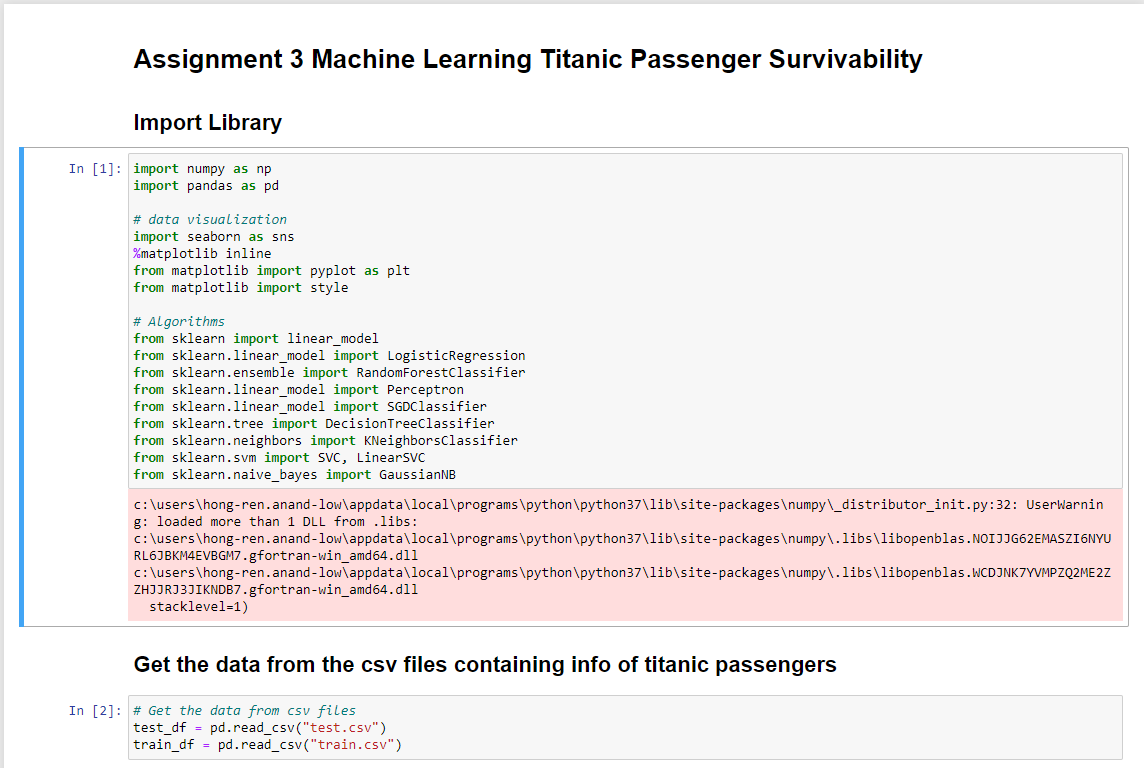


Figure: Titanic passenger survivability source code

# Assignment 3 Cat vs Dog Neural Network Classifier

### Getting the Data

The image dataset is taken from Kaggle machine learning competition held in 2013. The image dataset was supposed to be used as a standard computer vision dataset that involves classifying photos as either containing a dog or cat. The dataset consists of photos of dogs and cats provided as a subset of photos from a much larger dataset of 3 million manually labelled photos. The dataset consists of 12,500 dogs and 12,500 cats photos. The dataset was developed as a partnership between Petfinder.com and Microsoft.

Solving this problem may sound simple to humans, but for computer vision and machine learning, it was only effectively addressed in the past few years using deep learning convolutional neural networks. The dataset was initially planned to be used as a CAPTCHA to distinguish between human users and bots. However in 2007 a machine learning model built using SVM was able to achieve 80% classification accuracy, which voided the usability of the task for CAPTCHA.

Data Visualization

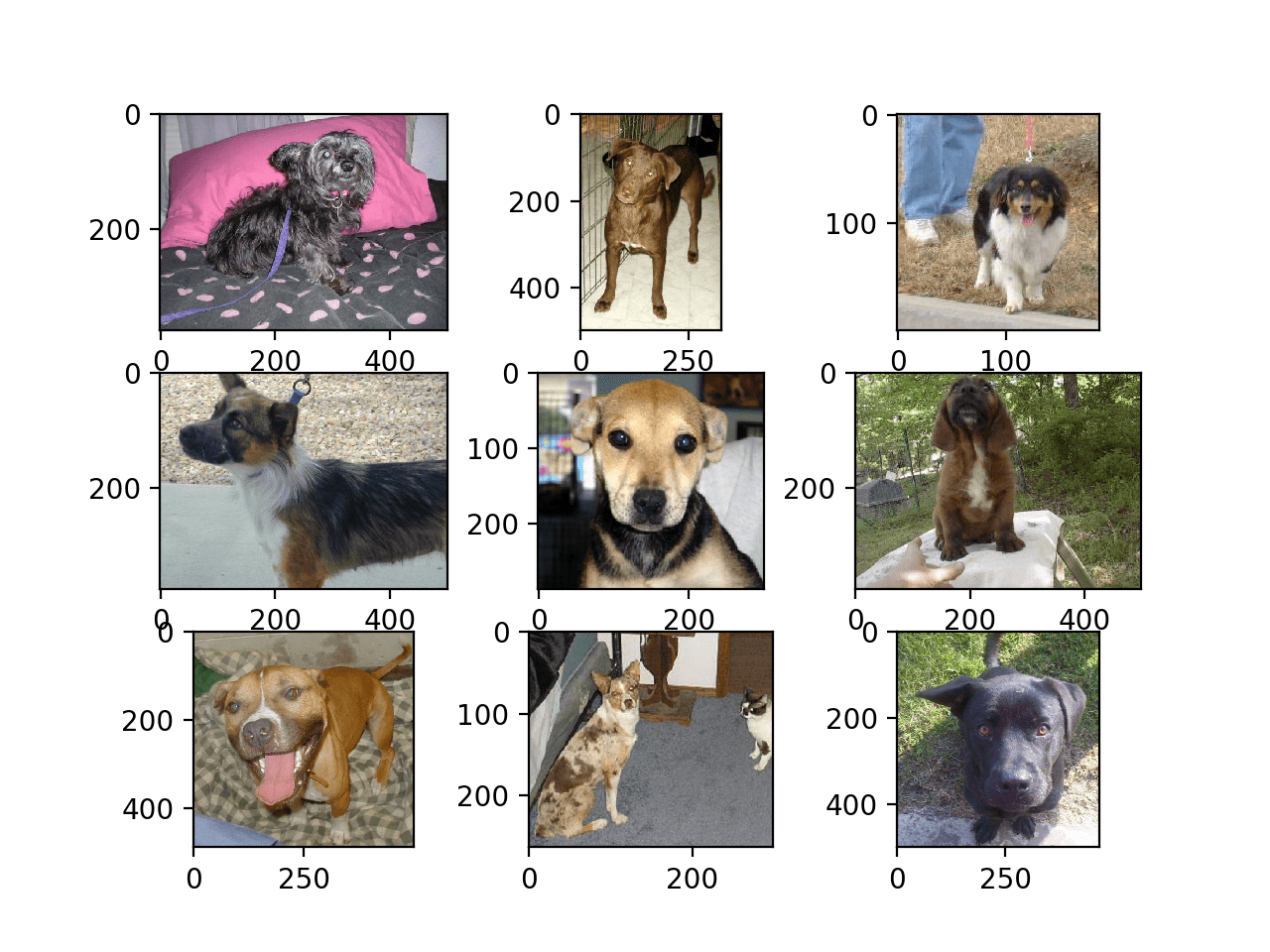


Figure: Some examples of the dog photos in the dataset

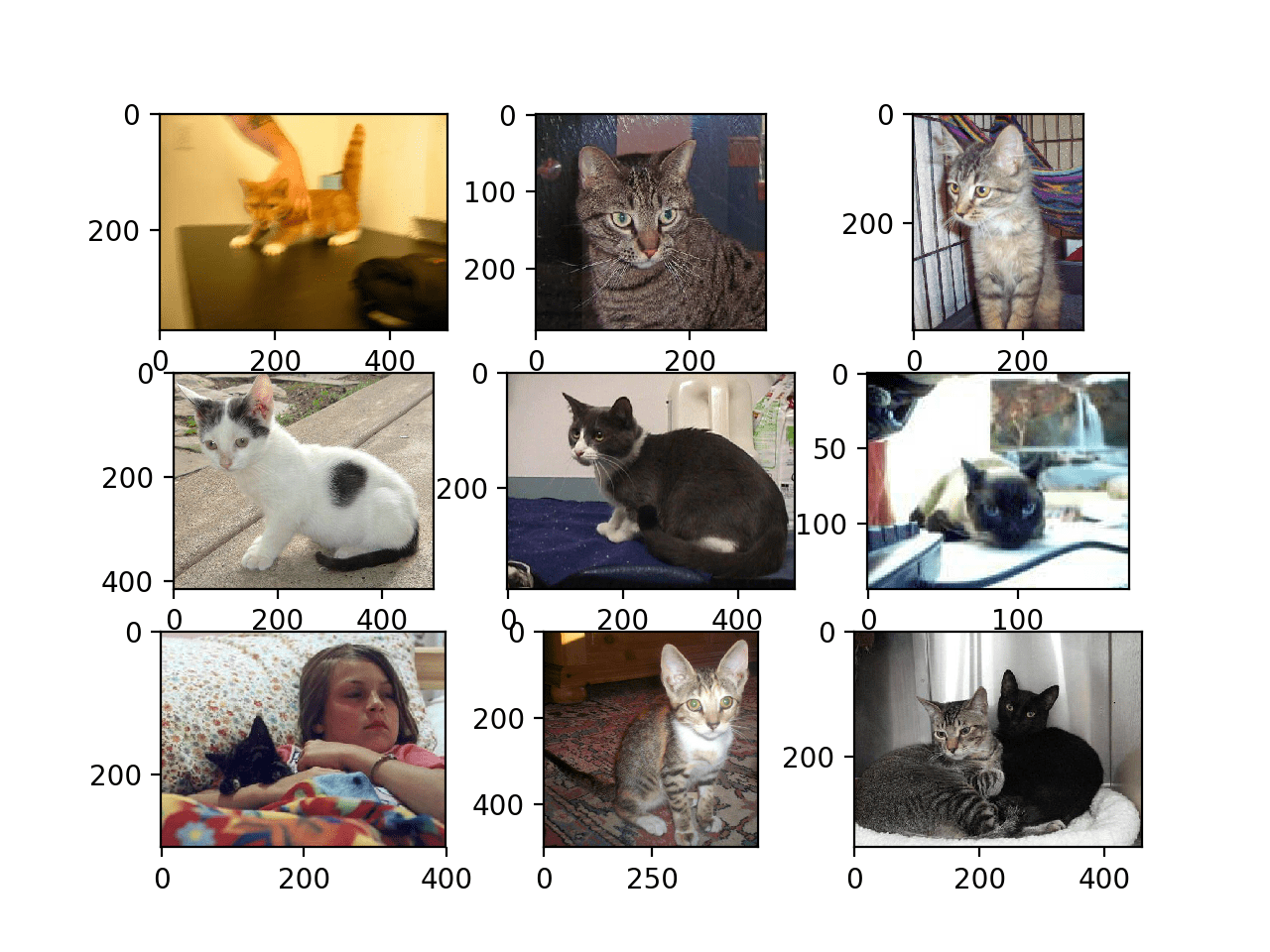


Figure: Some examples of cat photos in the dataset

We can see that some photos are in landscape format, while some are in portrait format, and some are square. Each photo has different dimensions. We can also see that the dataset includes photos with variations such as having 2 cats ( lower right) and a girl holding a not so visible cat (lower left). This suggests that any classifier fit on this problem will have to be robust.

### Distribute Training Images

Instead of writing a Python program to read the files from the hard disk, I use ImageDataGenerator from Keras library. This class can load images from the hard disk and generate batches of image data that the training process can use directly. The class will structure the data a certain way on the disk. For each label or category, I have to create a subdirectory with the same name as the label, as shown below.

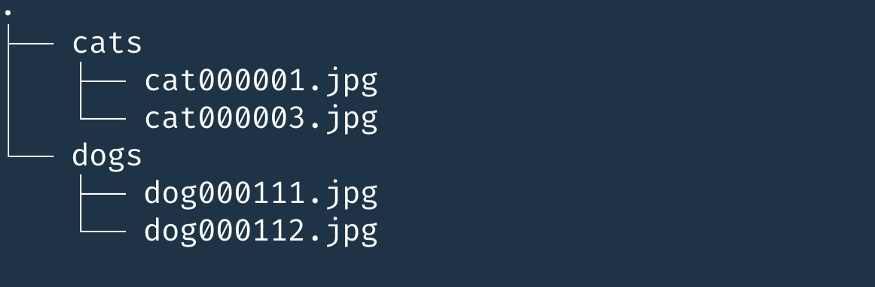


Figure: Data Structure of the files

Next, we also have to separate the files into training and validation datasets. In this assignment, I split the data into 75% training dataset and 25% validation dataset.



Figure: Distribute data into training and validation datasets

### Data Preprocessing

Since our photos come in different sizes, the first thing we have to do is standardize all the photos to one size. I decided to set the image width and height to 150 pixels respectively. This gives the image a dimension of 150x150 px. The RBG aspect of the image will be retained as well, but it is scaled down from 0 - 255 to 0 - 1. Choosing a smaller size like this will decrease the input load, which makes our neural network be able to train faster.

Since the photos are already manually labelled cat or dog following a number, we should be able to extract that information and divide our photos into two categories, 1 for dog and 0 for cat, since the neural network is only able to work with numbers.

* directory — The directory that contains the subdirectories with the categories.
* target\_size — Each image must be of the same size before submitting it to the deep learning pipeline. The ImageDataGenerator can perform this on the fly.
* batch\_size — We submit the images to the deep learning pipeline in batches. This sets the size of the individual batches.
* class\_mode — A hint of the type of data we read. In our case, ‘binary’ to show that we will use two categories.

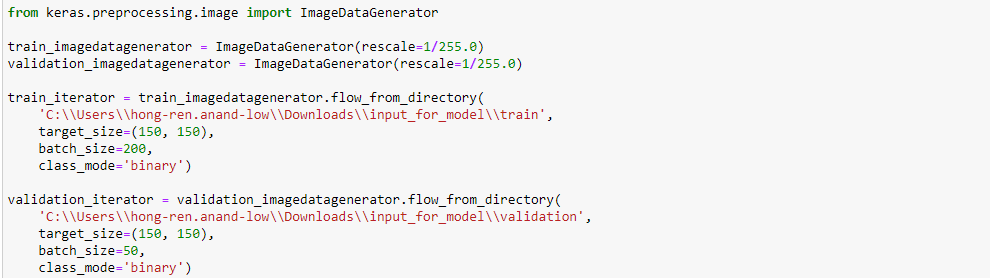


Figure: Rescale RGB value, set picture size, batch size and class mode



Figure: 18750 photos belonging to training dataset and 6250 photos belonging to validation dataset

### Create Neural Network Model

I researched online and chose to develop a baseline convolutional neural network model for the dogs vs cat dataset.

A baseline model is the first model that we try to establish a minimum performance which can be improved further through modification and study in the future. We split the model into three major parts. First, there are three combinations of the *Conv2D* and *MaxPooling2D* layers. These are called convolution layers. A *Conv2D* layer applies a filter to the original image to amplify certain features of the picture. The *MaxPooling2D* layer reduces the size of the image and reduces the number of needed parameters. Reducing the size of the image will increase the speed of training the network.

After that, we flatten the array, then create a hidden Dense layer with 512 units and use the Rectified Linear Unit (RELU) as the activation function.

The last part is the output layer. This layer has a single output neuron which outputs 0 (cat) or 1 (dog).

The model is then compiled.

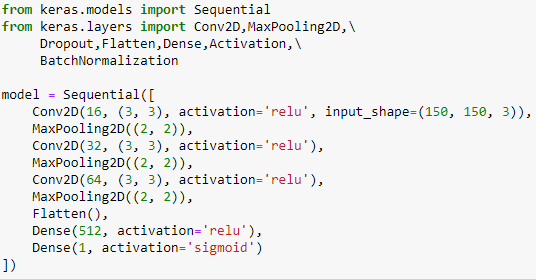


Figure: Build model

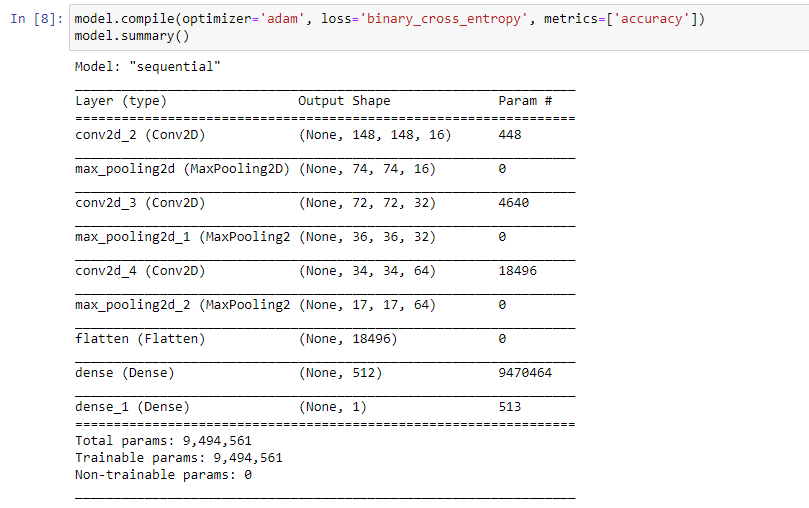


Figure: Compile model

Adam optimizer is an excellent default to start with. Binary Cross Entropy is chosen because the problem we are solving is a binary classification problem. And lastly the accuracy will be reported during training.

### Training the model

After we compiled the model, we can start training the model. We start training by calling the *fit* method on the model instance.

The first parameter is the iterator that we created by calling the *flow\_from\_directory* of the *ImageDataGenerator* for the training data. The second parameter is the iterator from the validation data.

The epoch parameter shows how many times the model will process the entire training set.

The steps\_per\_epoch shows how many batches it should process before it finishes the epoch. By multiplying a batch size of 20 and 1000 steps per epoch, we will get 20,000 which is the number of images of our training set.

Our validation steps will be 500 since 500\*10 = 5000, the number of images in our validation set.

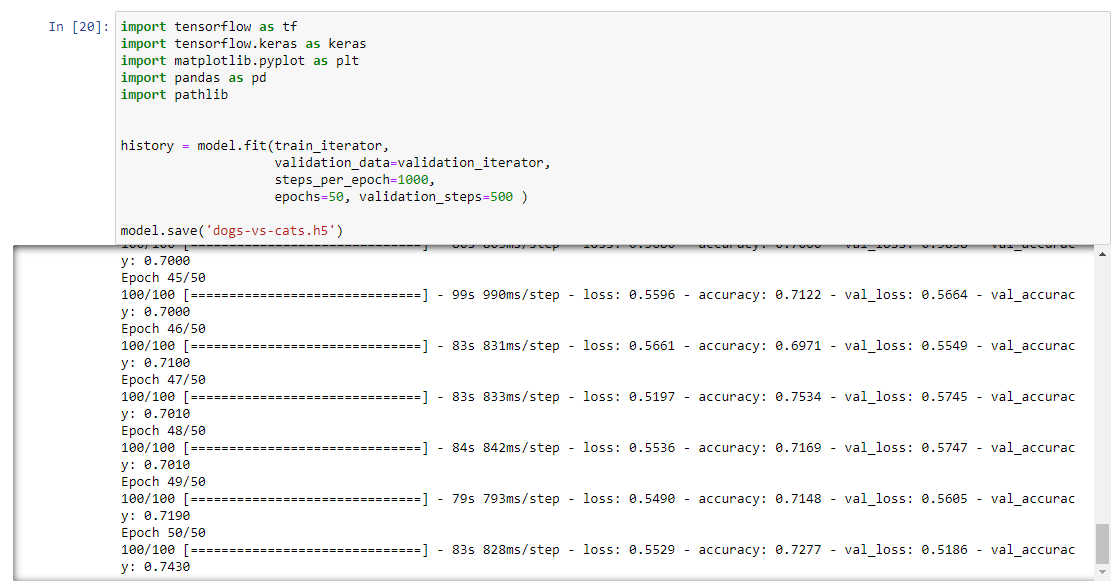


Figure: Training the model

### Results



Figure: Visualise the neural network training

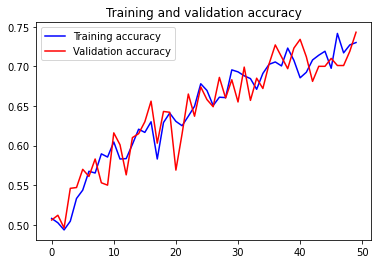


Figure: Training and validation accuracy

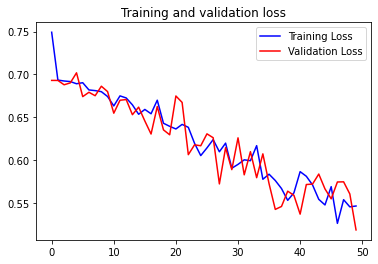


Figure: Training and validation loss

Both the training set’s accuracy and validation set’s accuracy increases from 0.5 to around 0.75. This means that there is no overfitting and the neural network is working fine as it is. Modification and optimization can be done to increase the accuracy further.