Evaluating the survival of passengers on the Titanic using Artificial Intelligence Algorithms

Report by

JAYSHIV PATEL

To accompany an artefact for the

LEVEL 3 EXTENDED PROJECT QUALIFICATION (EPQ)

Under the supervision of

MR SISULU

2020

Contents

Glossary

Artificial Intelligence (AI) - a type of computer technology which is concerned with making machines work in an intelligent way, similar to the way that the human mind works (Collins, 2019)

Gradient Boosting - a mathematical model that goes through cycles using **iterative methods** to add models into an ensemble to generate increasingly better predictions

Graphical User Interface (GUI) - a type of visual interface that allows the user to communicate with a computer, consisting of menus and icons that can be controlled by a mouse, for example, the Windows 10 Start Menu

Iterative methods - repeating a process to obtain successively closer approximations to the desired target

Object-Oriented Programming (OOP) - programming based around 'objects' as opposed to being based around functions (procedural)

Random Forest - a mathematical model that consists of many decisions trees (a model assessing decisions and their consequences) and uses a voting system to operate as an ensemble of many trees

Test Data - data that is used to test the accuracy and success of a model without the key parameter to predict its value

Train Data - data that is used to train a mathematical model which includes the key parameter of interest

Introduction

Computer Science has always been one of my passions, and one of the most exciting areas involves software development. This includes the decomposition of complex problems and planning their solutions. My inspiration for this EPQ artefact is to build upon my existing skills and to extend beyond the typical programming syllabus.

My report is centred around **artificial intelligence (AI)** supported by knowledge and a template offered by a specialist. The template outlined the production of an accurate and consistent machine learning model from start to finish as well as the use of two different models, discussed in detail later.

Many companies have begun using machine learning within everyday applications such as the cameras in Huawei smartphones to improve appearance or even Apple's Siri with its AI-powered speech recognition. These help to enhance the quality of life and are designed to make tasks easier. As a technology enthusiast, I was keen to implement a machine learning program in my own way. My inspiration for this project stemmed from seeing the impact of COVID-19 and made me question whether AI models could be used for safety and wellbeing. Following this idea, I decided to focus on transport, to test the use of AI to improve safety. To make it even more interesting, I chose the case of the Titanic, which is considered one of the most iconic forms of transport to date, using a free dataset (Kaggle, 2012).

The main idea was to create a baseline and later a more complex model, which allowed me to assess and evaluate their accuracy. Based on my analysis, the better model would be combined with a **graphical user interface (GUI)**. The final step was to then produce an artefact for a better insight on survival with regards to the Titanic.

Proposal and Planning

Initially, I intended to create an artificial intelligence algorithm that would be able to predict the survival of passengers on the Titanic based on their attributes. Throughout the process of the EPQ, this initial idea evolved into a GUI based system, to create a more well-rounded program [Fig. 1].

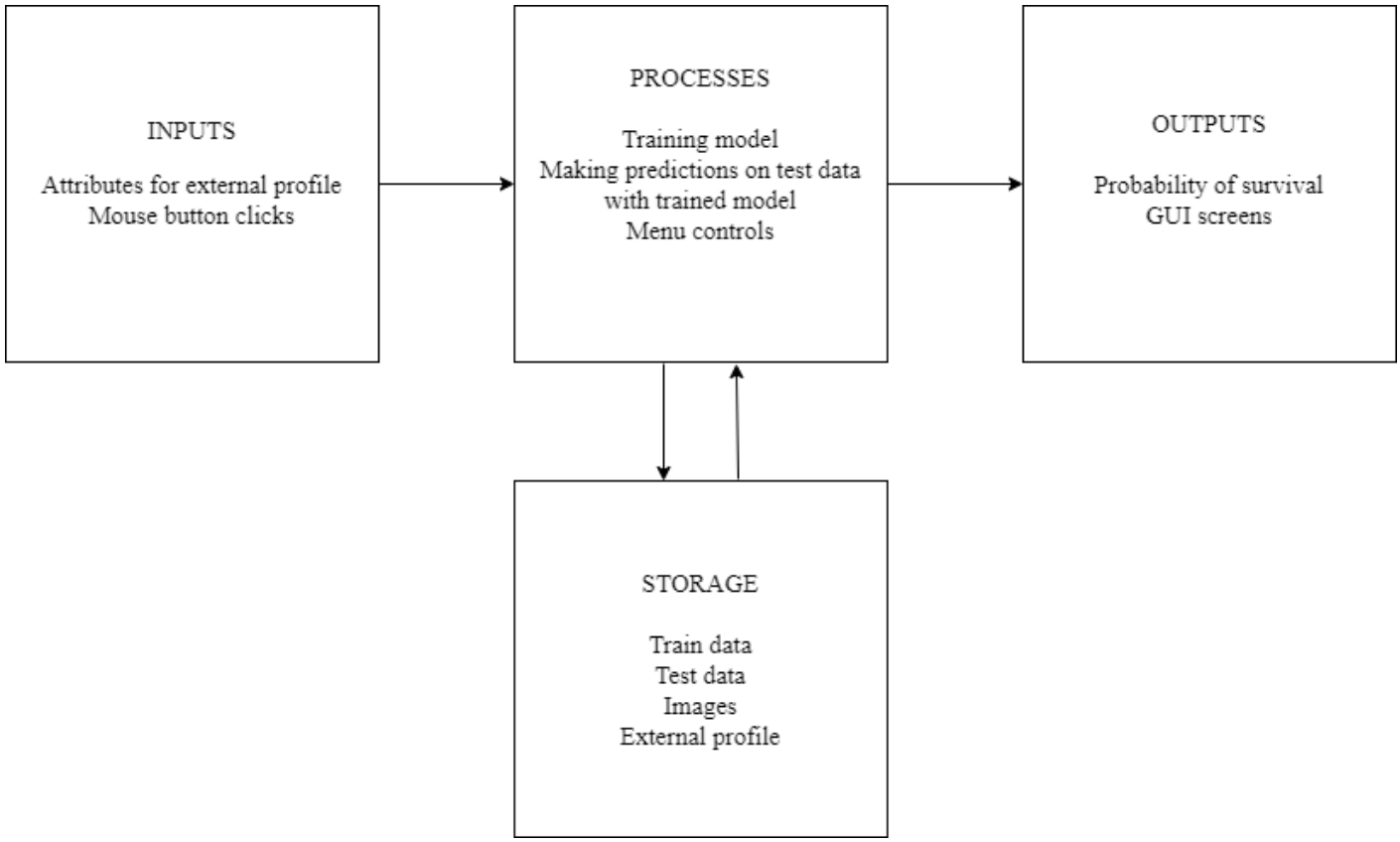

Figure 1: An input process storage output diagram of the intended product.

The success criteria for this program was as follows:

- The final model should be able to make reliable predictions about the survival of passengers on the Titanic given their profile, i.e. age, sex, class etc… [Fig. 2]
- The final model will be combined in conjunction with a basic system to allow for fiction/real profile selection/creation as the user desires, to allow the user to test whether profiles would have survived the sinking.
- The final model should be accompanied by a GUI.
- This will allow me to trial the modification of features of the passengers, e.g. what if passenger 15 was in 2nd class seating? Would they have survived? From here I will be able to see if certain changes are more/less likely to affect survival
- This will also allow me to work out the probability of survival based on given statistics.

ID	Survived	Class	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Port
164	0	3	Calic, Mr. J	male	17	0	0	315093	8.66		S
165	0	3	Panula, Ma	male	1	4	1	3101295	39.7		S
166	1	3	Goldsmith,	male	9	0	2	363291	20.5		S
167	1	1	Chibnall, M	female		0	1	113505	55	E33	S
168	0	3	Skoog, Mr	female	45	1	4	347088	27.9		S
169	0	1	Baumann,	male		0	0	PC 17318	25.9		S
170	0	3	Ling, Mr. L	male	28	0	0	1601	56.5		S
171	0	1	Van der ho	male	61	0	0	111240	33.5	B19	S
172	0	3	Rice, Mast	male	4	4	1	382652	29.1		Q
173	1	3	Johnson, M	female	1	1	1	347742	11.1		S
174	0	3	Sivola, Mr.	male	21	0	0	STON/O 2.	7.93		S
175	0	1	Smith, Mr.	male	56	0	0	17764	30.7	A7	C
176	0	3	Klasen, Mi	male	18	1	1	350404	7.85		S
177	0	3	Lefebre, M	male		3	1	4133	25.5		S
178	0	1	Isham, Mis	female	50	0	0	PC 17595	28.7	C49	C
179	0	2	Hale, Mr. F	male	30	0	0	250653	13		S

Figure 2: This is a sample of the data from the train.csv file provided by Kaggle. As highlighted in black, there are many gaps in this dataset, and fields such as ticket had to be omitted from the model because of the inconsistencies with regards to the formats provided.

Research and Development

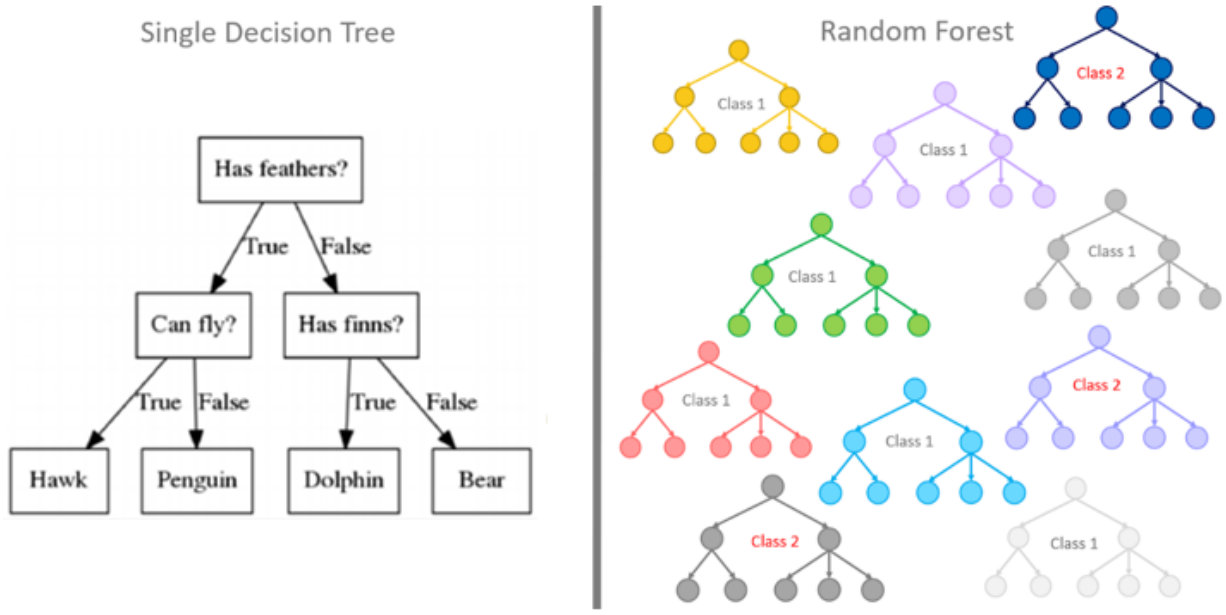

Figure 3: A breakdown of the Random Forest model, showing how it utilises multiple decision trees together as an ensemble rather than as a standalone tree (Chodwary, 2020; Silipo, 2019).

My project started with research into AI, from watching documentaries to reading a variety of articles, and machine learning courses (DeepMind, 2020; Hardesty, 2017; Becker, 2019; Cook, 2020; Goyal, 2020). I compared some of the most commonly used models (Gupta, 2020; Kho, 2018; Singh, 2018; Yiu, 2019; Brid, 2018). Of these, I chose '**Random Forest**' and '**Gradient Boosting**' as my baseline and complex models respectively; the former from the 'scikit-learn' python library and the latter from the 'XGBoost' library. I decided to use the 'scikit-learn' library because of its beginner-friendly nature and the 'XGBoost' library because of its more complex but more powerful 'Gradient Boosting' model. 'Random Forest' involves using multiple decision trees that operate together to generate predictions [Fig. 3] while 'Gradient Boosting' involves iteration to build upon an original ensemble to try and minimise losses[Fig. 4].

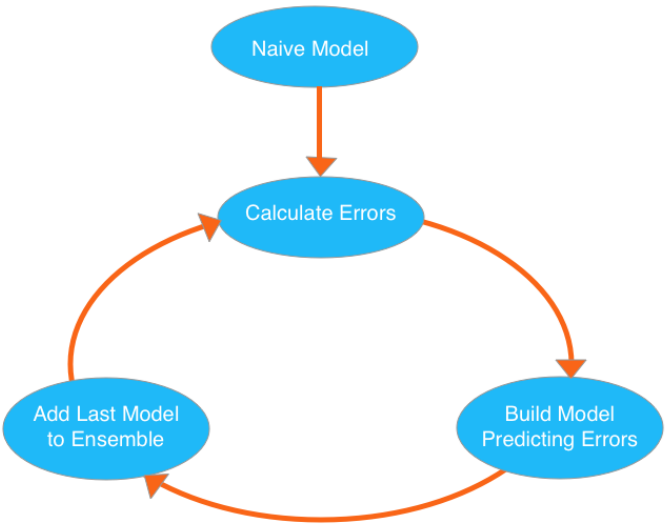

Figure 4: A breakdown of the 'Gradient Boosting' model showing the looping process which it uses to build its ensemble based on its errors in testing, which makes it usually more accurate but requires more parameters and tweaking (Becker, 2018).

When making the initial baseline model, I used techniques acquired from reading through the courses. Initially, I had to drop many entries for having some empty sections and had to modify the dataset myself. This meant that while the model could make predictions, it was highly dependent on my changes.

Some of my background research on the Titanic relates to the cabin locations of those who have died and survived (Lee, 2014). I tried to find a relationship between positioning and survival; however due to the limited number of passengers with registered cabins, I could find no discernible trends. Although, I did notice a very high survival rate specifically on Deck D. This could be since most of the cabins on this deck were first class and included higher-paying passengers.

Further research involved looking at a variety of examples of machine learning code (Anon, 2020); Shukla, Iriondo and Chen, 2020). This step allowed me to select some of the ideas, implemented later into my program. The machine learning code together with the knowledge obtained from courses were necessary to build upon my original code. Once the baseline model was created with the addition of the 'accuracy_score' feature I was able to check the accuracy of its predictions on **test data** (after splitting up the **train data**) and produce a program that could manipulate data on its own. By using **iterative methods** [Fig. 5] to tweak the settings, I was able to put my original model against itself while improving the initial accuracy.

```
#We can see that increasing the number of nodes past 25 has almost no effect,
#and peak accuracy occurs at around 10 nodes.
#Here I have rerun the iteration with a smaller scope around 10 nodes
def accuracy_of_setting(max_leaf_nodes, train_X, valid_X, train_y, valid_y):
    baseline_model = RandomForestRegressor(max_leaf_nodes = max_leaf_nodes, random_state = 1)
    baseline_model.fit(train_X, train_y)
    predicted_values = baseline_model.predict(valid_X)
    rounded_values = []
    for item in predicted_values:
        item = round(item, 0)
        rounded_values.append(item)
    return(accuracy_score(rounded_values,valid_y)*100)


results = {}
for max_leaf_nodes in range(5,15):
    model_accuracy = accuracy_of_setting(max_leaf_nodes, OH_X_train, OH_X_valid, train_y, valid_y)
    results[max_leaf_nodes] = model_accuracy
    print('Max leaf nodes: %d \t\t Accuracy %d' %(max_leaf_nodes, model_accuracy))
```

```
Max leaf nodes: 5               Accuracy 82
Max leaf nodes: 6               Accuracy 82
Max leaf nodes: 7               Accuracy 82
Max leaf nodes: 8               Accuracy 82
Max leaf nodes: 9               Accuracy 82
Max leaf nodes: 10              Accuracy 82
Max leaf nodes: 11              Accuracy 81
Max leaf nodes: 12              Accuracy 81
Max leaf nodes: 13              Accuracy 82
Max leaf nodes: 14              Accuracy 81
```

```
#I will take 8 to be the optimimum number of leaf nodes, slightly improving the accuracy of my
#baseline model after use with imputation.
#I have now improved the accuracy of the initial version by around 6%!
```

Figure 5: An example of using iterative methods to improve the accuracy of the Random Forest model, showing how changing a feature such as the maximum number of leaf nodes, accuracy of the model can change.

Accuracy was further improved when I used the complex model and tweaked it using the same methodology of using iteration. The model reached a peak accuracy of 86%, [Fig. 6] a vast improvement over the original baseline. The next steps were to plan and design my GUI on paper before coding this into the AI portion to complete the artefact. To assist with making the GUI, I looked at some specific examples of code (McGrath, 2013; Amos, 2020) and followed general good-practice methods while code-writing (Long, 2017).

```
processed_X_train = pd.concat([imputed_X_train.reset_index(), OH_cols_train.reset_index(
processed_X_valid = pd.concat([imputed_X_valid.reset_index(), OH_cols_valid.reset_index(

processed_X_train = processed_X_train.drop(['index'], axis =1)
processed_X_valid = processed_X_valid.drop(['index'], axis =1)

processed_X_train = processed_X_train.apply(pd.to_numeric)
processed_X_valid = processed_X_valid.apply(pd.to_numeric)

#The complex model
model = XGBRegressor(max_depth = 5, n_estimators = 1000, learning_rate = 0.19,
                     objective = 'reg:squarederror')
model.fit(processed_X_train, train_y, early_stopping_rounds = 5,
                      eval_set=[(processed_X_valid, valid_y)], verbose = False)
predicted_values = model.predict(processed_X_valid)
rounded_values = []

for item in predicted_values:
    item = round(item, 0)
    rounded_values.append(item)
accuracy = (accuracy_score(rounded_values,valid_y)*100)

print('Accuracy = ' , accuracy)
```

```
Accuracy =  86.09865470852019
```

Figure 6: This is a snippet of the complex model after rigorous testing and improvements like the method used in Figure 6. Here, the model reaches its highest accuracy of 86%. See Appendix for Final Product.

Evaluation and Conclusion

The 'Gradient Boosting' model showed that ticket fare was the most significant factor it considered when calculating the probability of survival [Fig. 7]. This order was reinforced when I was able to re-enter existing profiles while changing certain features, e.g. to have a higher ticket fare or have an age within the range of 10 – 25. This was done while keeping other attributes constant and consistently returned a significantly higher probability of survival.

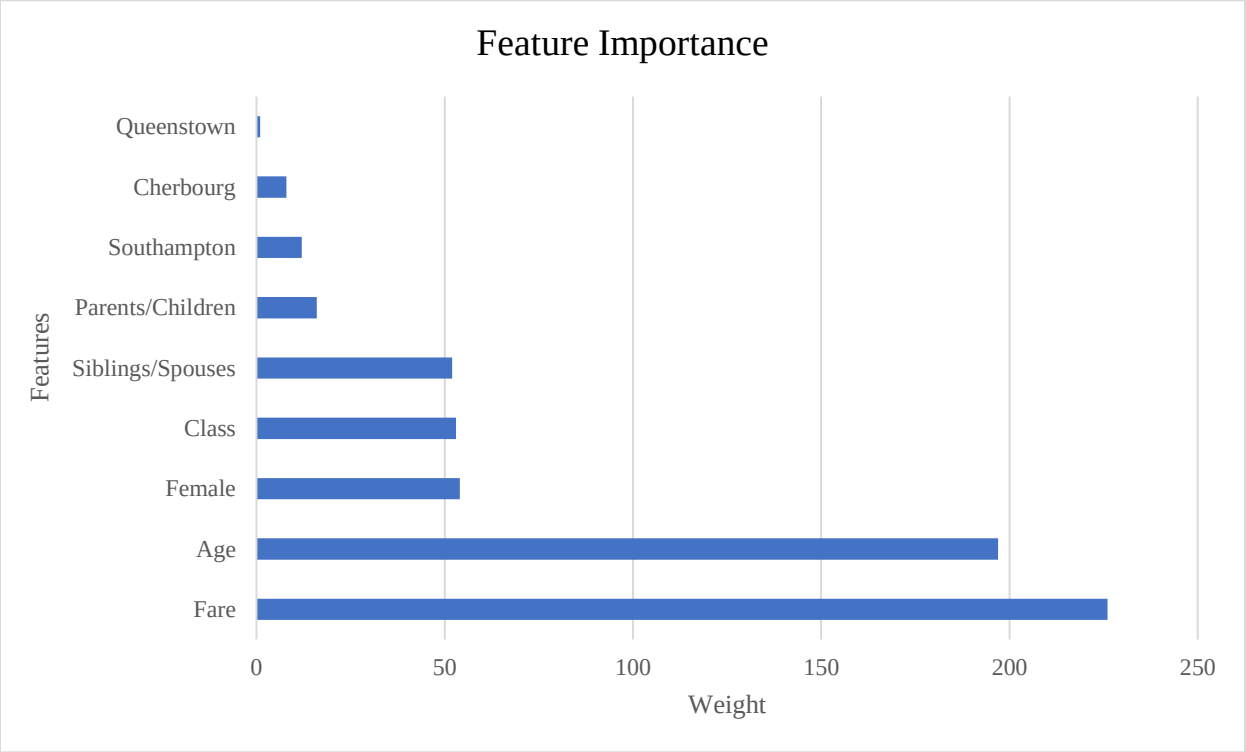

Figure 7: Graph of F-score (feature importance) output by the complex model. This shows the importance of all the features with relation to each other and how much weight they add to the model.

Based on my initial background research on the Titanic (Anon, 2015; Bolan, 2012a; Bolan, 2012b; Simple History, 2018; Smith, 2017; Tikkanen, 2018), I had initially hypothesized that class (1st ,2nd , or 3rd) would be the most prominent factor that determined survival, given that the ship had a reduced number of lifeboats aboard and that the upper class had an advantage in boarding these boats. Since fare and class are related, the former's appearance at the top of the chart is justified. Additionally, the protocol at sea at the time had been that lifeboats would be loaded with women and children first, explaining why being female is the third-highest feature.

Two of the more interesting attributes that I did not initially anticipate would appear were age and sibling/spouse count (SibSp) coming 2nd and 5th respectively in weight. Perhaps age was so relevant (10-25 found in my testing as the best age range) because adolescents were more likely to have a place on the lifeboats and young adults were able to survive for longer in the sea. SibSp could be as prevalent as it is because larger families could work together to make it to a lifeboat and were more likely to stay alive.

Today, transport is ubiquitous and takes many forms. Sadly, fatalities still occur, and this can be problematic when at sea/in the air where emergency services cannot be omnipresent. My artefact is specific to the events of the Titanic, which occurred over 100 years ago and is unlikely to be accurate for an event today, like a long-distance flight.

However, my approach using machine learning models to identify trends with those who have lost their lives could be used by transport companies to then implement safety features tailored to those at a higher risk of death. Combining the models with a GUI means that the information is simple to follow and understand by transport staff, thus allowing for airlines/ships to be better prepared. For example, if a flight company uses its own database to train a successful AI model on data from previous accidents/injuries, passenger information could be checked upon booking their tickets. If a passenger has multiple flagged features by the models created, they could be highlighted by onboard staff, and be given priority in the event of an evacuation. This then ensures that those at higher risk are better prepared.

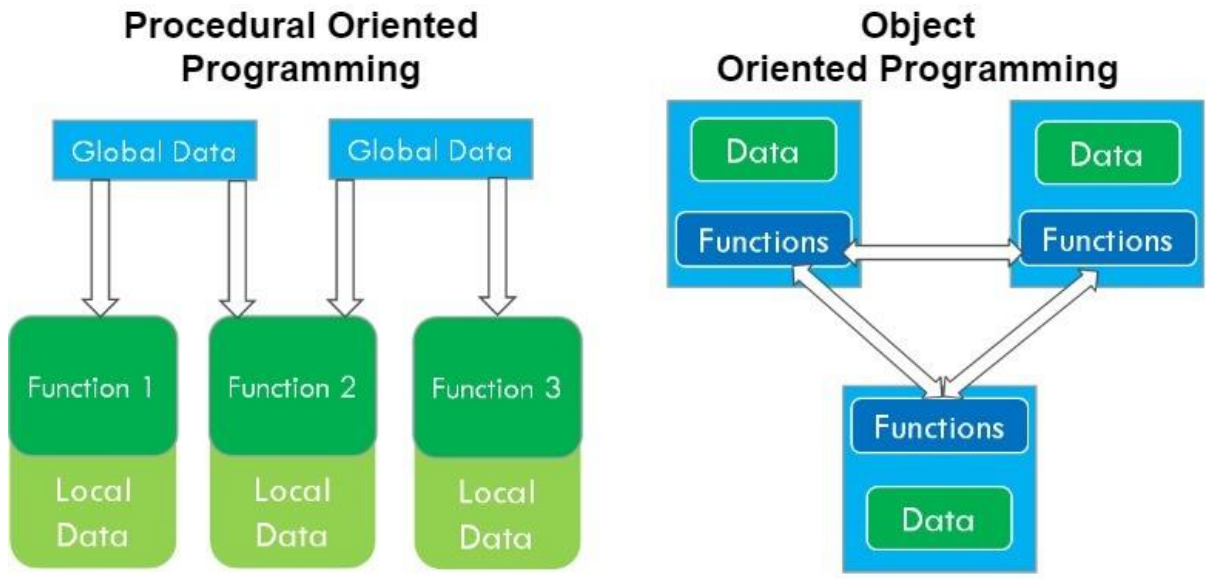

Figure 8: This image helps to explain why OOP may have reduced the complexity of the code. As we can see from the left portion of the image, procedural programming relies on functions sharing global data but are not able to share local data. OOP (on the right) does not suffer from this problem, as it allows code to be inherited, making it more efficient and better abstracted. (kkris, 2020).

Having many dependent functions, which meant that the code became convoluted, was one of the challenges related to the production of my artefact. To overcome this, one could have completed additional research into **object-oriented programming (OOP)** [Fig. 8]. This could encapsulate and help abstract the code and reduce its complexity.

Another challenge I approached during my work was the dataset itself. There were many missing pieces of information like the cabin placements and ticket numbers. An improvement would require repeating the process of analyses by using a dataset form from a different source with more available information than Kaggle – however, this could be costly. Furthermore, I had an issue with running the final code on my current computer (Windows 32-bit) due to incompatibility. This means that the screenshots of the final interface appear not as expected (taken while running on a MacBook) while the designs were done on Windows and so look slightly different. Coding the separate models (baseline and complex) as well as the GUI were very time-consuming. It would have been possible to have simply coded one model, and not included the ability of the program to clean the data itself. However, I feel that I have been able to create a better-fitted model by pitting the two against each other and have produced a more flexible code.

Working on this project provided me with an opportunity to improve my programming skills, research, and organisational skills. Moreover, I managed to achieve all the set tasks over a short period. My code was also peer-reviewed by a Cambridge computer science student with the resultant implementation of some changes.

Overall, I have been able to meet all my success criteria when completing the final project, and while I did not stick to my original time plans, was able to manage tasks effectively. With regards to my working practice, I feel that my subsequent time management and comprehensive documentation has been consistent and well-kept. I am pleased with the outcome of this project and believe that it has been successful.

Acknowledgements

I would like to give thanks to Mr Sisulu for accepting to supervise this EPQ and for offering his knowledge and advice. Additionally, to Ms Patel for her help and guidance during the lockdown period, Dr Dusza for his suggestions and finally to Dr Chikvaidze, for providing her assistance.

References

Amos, D. (2020). Python GUI Programming with Tkinter – Real Python. [online] realpython.com. Available at: https://realpython.com/python-gui-tkinter/.

Anon (2015). RMS Titanic facts. [online] Royal Museums Greenwich. Available at: https://www.rmg.co.uk/discover/explore/rms-titanic-fact-sheet.

Anon (2020). First Machine Learning Project in Python Step-By-Step In 2019 (Updated) | FavouriteBlog.com. [online] Favourite Blog. Available at: https://favouriteblog.com/first-machine-learning-project-in-python-step-by-step/.

Becker, D. (2018). XGBoost. [online] Kaggle.com. Available at: https://www.kaggle.com/dansbecker/xgboost.

Becker, D. (2019). Intro to Machine Learning. [online] Kaggle.com. Available at: https://www.kaggle.com/learn/intro-to-machine-learning.

Bolan, P. (2012a). Sinking of the Titanic. [online] National Geographic Society. Available at: https://www.nationalgeographic.org/media/sinking-of-the-titanic/?utm_source=BibblioRCM_Row.

Bolan, P. (2012b). The R.M.S. Titanic. [online] National Geographic Society. Available at: https://www.nationalgeographic.org/media/rms-titanic-photo-gallery/#:~:text=The%20Titanic%20sailed%20out%20of.

Brid, R.S. (2018). Decision Trees—A simple way to visualize a decision. [online] Medium. Available at: https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb.

Chodwary, D. (2020). Decision Trees Explained with a Practical Example. [online] towardsai.net. Available at: https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53.

Collins (2019). Artificial intelligence definition and meaning | Collins English Dictionary. [online] Collinsdictionary.com. Available at: https://www.collinsdictionary.com/dictionary/english/artificial-intelligence.

Cook, A.B. (2020). Learn Intermediate Machine Learning Tutorials. [online] www.kaggle.com. Available at: https://www.kaggle.com/learn/intermediate-machine-learning.

DeepMind (2020). AlphaGo - The Movie | Full Documentary. YouTube. Available at: https://www.youtube.com/watch?v=WXuK6gekU1Y.

Goyal, K. (2020). Machine Learning vs Neural Networks: What is the Difference? [online] upGrad blog. Available at: https://www.upgrad.com/blog/machine-learning-vs-neural-networks/#:~:text=While%20a%20Machine%20Learning%20model.

Gupta, S. (2020). Pros and cons of various Classification ML algorithms. [online] Medium. Available at: https://towardsdatascience.com/pros-and-cons-of-various-classification-ml-algorithms-3b5bfb3c87d6.

Hardesty, L. (2017). Explained: Neural networks. [online] MIT News. Available at: http://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

Kaggle (2012). Titanic: Machine Learning from Disaster | Kaggle. [online] Kaggle.com. Available at: https://www.kaggle.com/c/titanic/data.

Kho, J. (2018). Why Random Forest is My Favorite Machine Learning Model. [online] Medium. Available at: https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706.

kkris, arvindpdmn (2020). Object-Oriented Programming Concepts. [online] Devopedia. Available at: https://devopedia.org/object-oriented-programming-concepts.

Lee, P. (2014). The Collision. [online] Paullee.com. Available at: http://www.paullee.com/titanic/thecollision.php.

Long, P. (2017). The Ultimate GCSE CS Textbook for AQA. paullong.net.

McGrath, M. (2013). Python. Leamington Spa, England: In Easy Steps.

Shukla, P., Iriondo, R. and Chen, S. (2020). Machine Learning Algorithms For Beginners with Code Examples in Python. [online] Medium. Available at: https://medium.com/towards-artificial-intelligence/machine-learning-algorithms-for-beginners-with-python-code-examples-ml-19c6afd60daa.

Silipo, R. (2019). From a Single Decision Tree to a Random Forest. [online] Medium. Available at: https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147.

Simple History (2018). Sinking of the Titanic (1912). YouTube. Available at: https://www.youtube.com/watch?v=b0L_2jKEbA4.

Singh, S. (2018). Cousins of Artificial Intelligence. [online] Medium. Available at: https://towardsdatascience.com/cousins-of-artificial-intelligence-dda4edc27b55.

Smith, O. (2017). Titanic: 40 Fascinating Facts. [online] The Telegraph. Available at: https://www.telegraph.co.uk/travel/lists/titanic-fascinating-facts/.

Tikkanen, A. (2018). Titanic | History, Sinking, Rescue, Survivors, & Facts. In: Encyclopædia Britannica. [online] Available at: https://www.britannica.com/topic/Titanic.

Yiu, T. (2019). Understanding Random Forest. [online] Medium. Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Appendix

Final Product - GUI

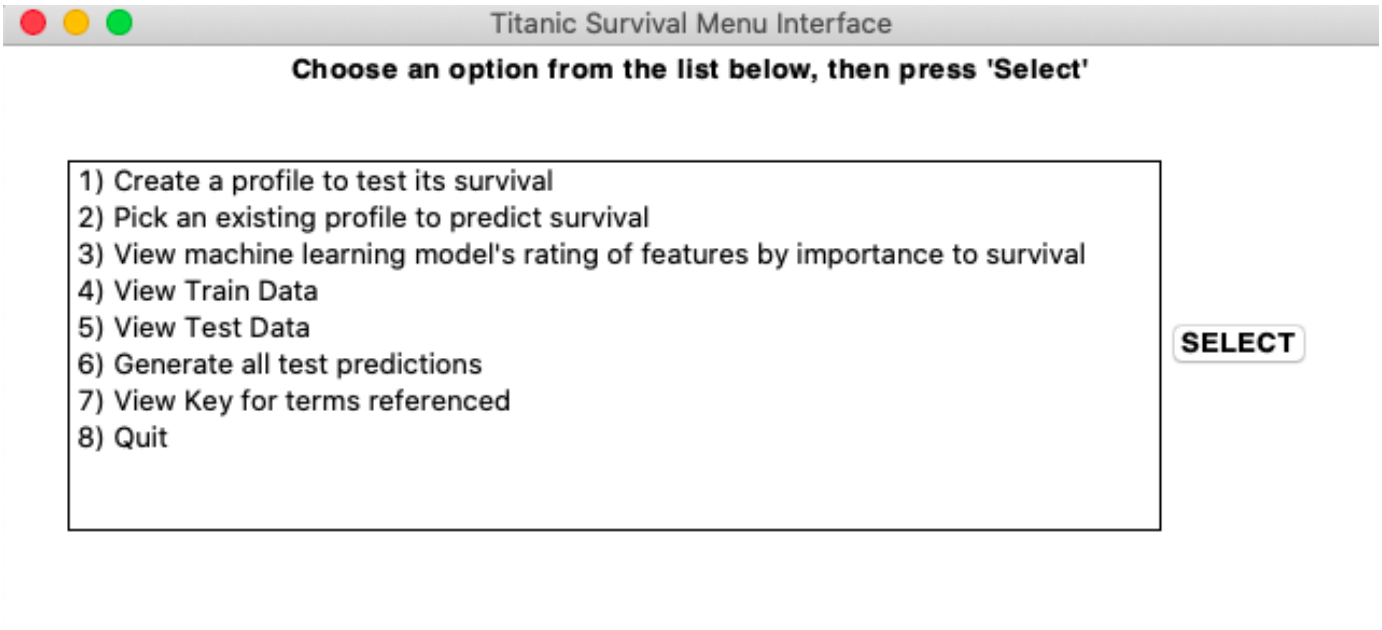

Figure 9: This screenshot shows the main menu, including a list box with a button to the right, allowing the user to select one of the eight options to proceed through the system.

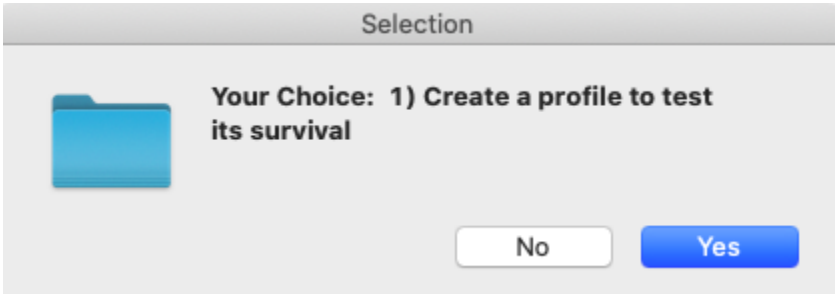

Figure 10: This is an informational screen, to confirm the choice selected from the menu.

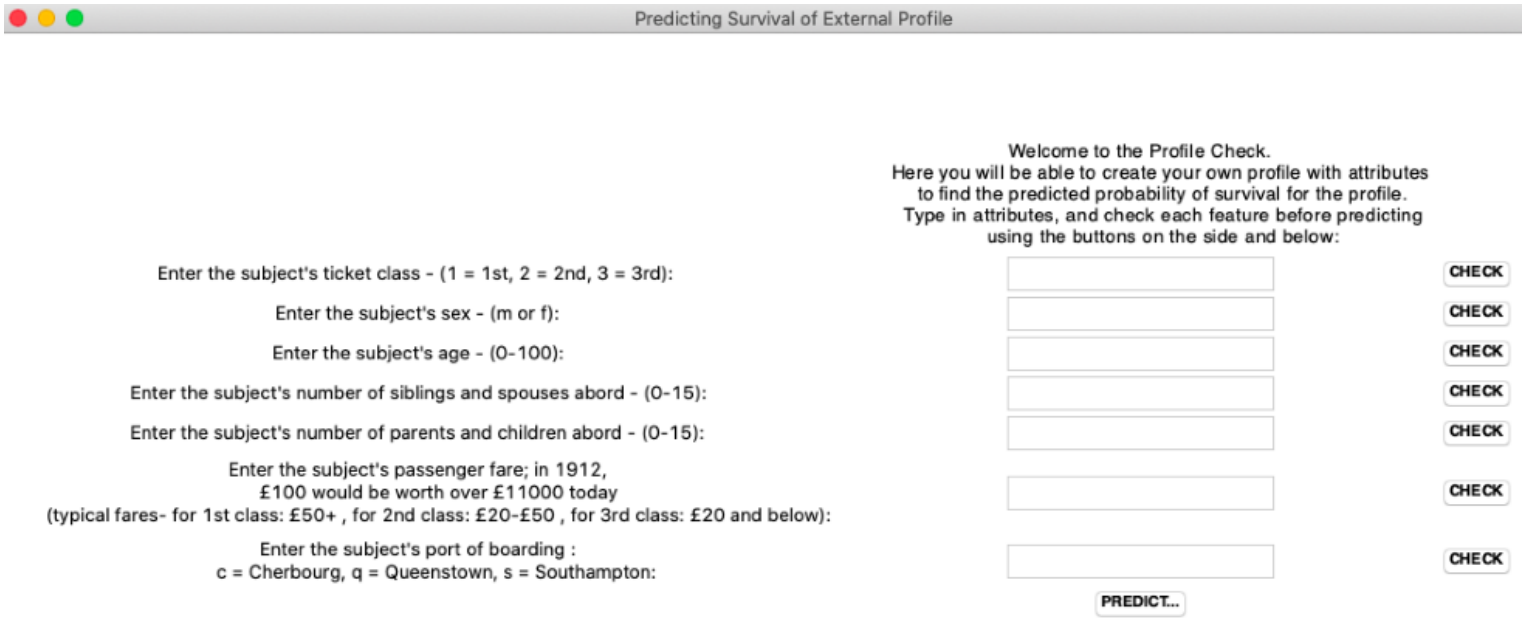

Figure 11: This is a screenshot of the profile prediction page, where custom attributes can be entered to check the survival of an external profile.

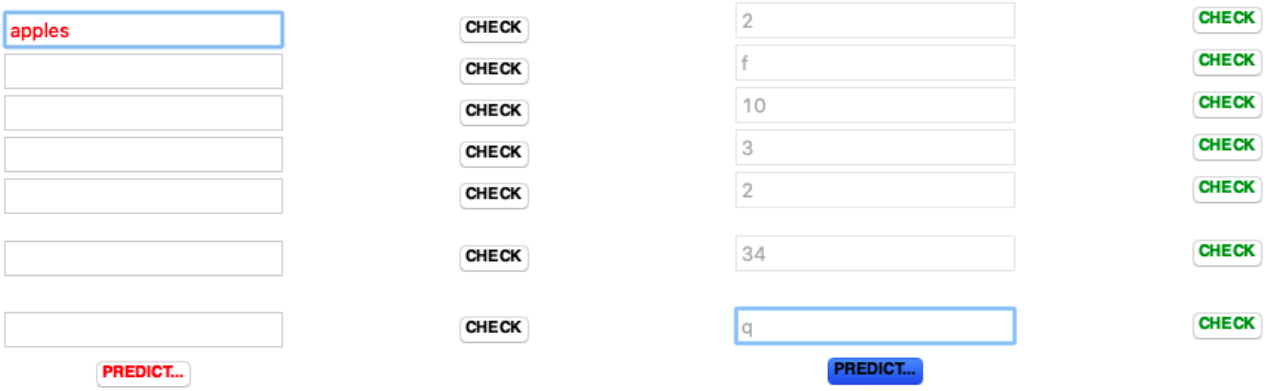

Figure 12 (sample of screen in Figure 9): Here you can see that each check field has its own validation routine, so that rogue data cannot make its way through the program and cause errors. This means that any data that does not match the intended format will turn red and will have to be re-entered correctly. In this case 'apples' is not accepted for the class (1, 2 or 3 would be acceptable) and as such with empty fields, a prediction cannot take place.

Figure 13 (sample of screen in Figure 9): This shows that with all entries checked (the buttons change to green) they cannot be changed, and the system allows for a prediction to take place.

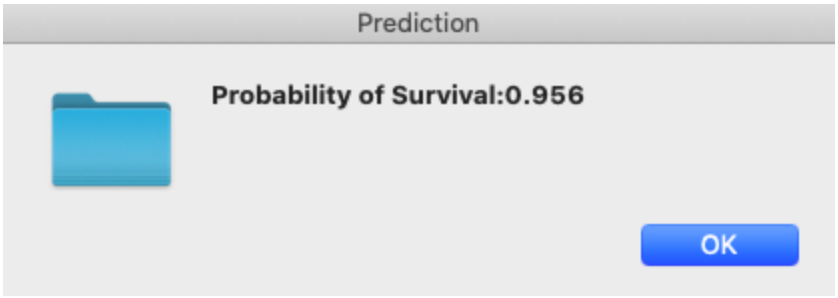

Figure 14: This informational pop-up screen shows the probability of survival that the model outputs as a separate screen for emphasis.

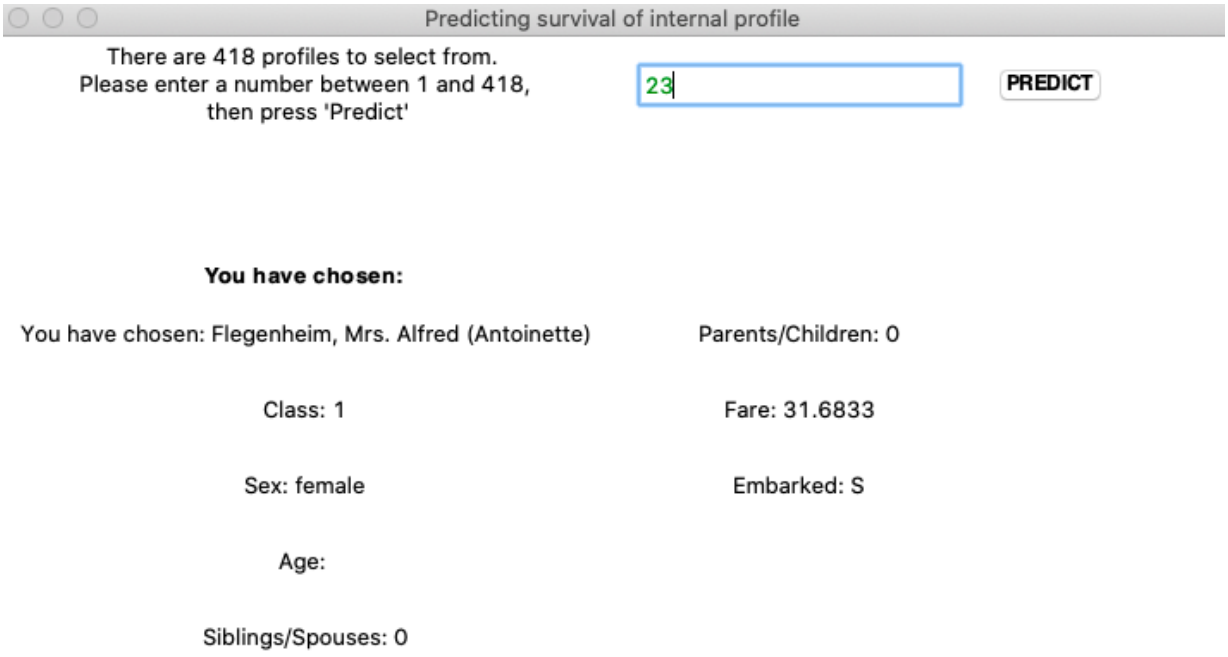

Figure 15: This screen allows the user to choose any of the 419 inbuilt profiles (included with the test data from Kaggle) that the model has not been exposed to before in order to predict their probability of survival. In this example we can see that we do not have data for the age in this record. However, behind the scenes, the program uses imputation to imitate the most popular value of that dataset to avoid crashes.