University of
St Andrews

CS4098: Senior Honours Project

Topic: Analyzing How Humans Make Material Decisions

Date of Submission: Spring 2021

Author: 180014200

Word Count: 17, 580

Credits: 15

Lecturer: Dr. Ognjen Arandjelović

Abstract

The study of human behaviour is one that has the most pervasive impacts on every aspect of our lives. This investigation of human behaviour has been instrumental for the conception of the *rational consumer* - a consumer that always behaves logically, and always seeks to maximise their own gains. As academia has largely used this theory of the *rational consumer* across a variety of different domains, there has always been some limitations to the capabilities of modelling, when aiming to capture scenarios that are heavily influenced by human nature. This is because human behaviour cannot be modelled perfectly linearly, and as such, deviations from expected rationale must be accounted for. Thus, being able to gain insight into the motivations that govern human behaviour is invaluable for academic fields such as economics, politics, computer science, etc.

This project aims to explain some of the motivations of contestants on the Australian television show, *The Chase*, such as the rationale behind their decision to choose one of the three offers presented (high, middle,low) the *Chaser*'s reasoning behind the (seemingly) arbitrary high and low offers. Alongside this, the project also examines the game environment of the show, in order to further consider and evaluate the different parameters that influence the game-play, such as the order of contestants, the amount of money contributed to the pot, etc.

We undertake this investigation through the use of *image processing* and *computer vision*, which allows us to analyze videos of *The Chase Australia*, and extract the essential information that we can conduct our data analysis on.
The *image processing* aspect of this project is conducted using *Python* and the *Open-CV* library, which allows us to analyse every $n^{t\ h}$ frame of a video, to deduce relevant information.
The data analysis aspect of this project is conducted using *pandas, numpy, scikit, sklearn, statsmodels* and *matplotlib*, which are all Python libraries that can be utilized to manipulate data.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 17.580 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bonafide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Acknowledgements

I would like to thank my supervisor, Dr. Ognjen Arandjelović for his continued support and guidance throughout the duration of this project. His advice has been invaluable during these challenging times, and our discussions have had a profound impact on this venture.

I would also like to thank the School of Computer Science for providing me with the opportunity to conduct independent analysis into such an interesting topic, and for the provision of 24/7 Jack Cole Labs - the perfect place to analyse hundreds of videos of The Chase.

Contents

1 Introduction

The popular Australian game show, *The Chase* is the primary environment that this project's analysis will focus on.

1.1 Motivation

1.1.1 Human Behaviour

When undertaking a project that explores human rationale, it only seems appropriate to explain our rationale behind the undertaking of this project. The key question is simple - *how do humans make decisions?*

This is a question that has remained unanswered for centuries. The lack of clarity on this matter has spurred the creation of a *rational consumer* in a number of social sciences, such as Economics. This concept of a *rationality* is used across economic models, extensively, in order to predict the effects of human behaviour on the phenomenon in question, and to predict how humans may behave in certain situations.

However, this postulation of rationality has garnered widespread criticism, as the assumptions made for rational decision-making is born from irrational expectation. Assumptions such as the rational consumer having access to *all* information that has ever existed in a relevant domain, having no other motivations other than pure self-interest, are always able to correctly profit maximise, etc, are examples of why consumers cannot be rational across every facet of life. Thus, the inclusion of this *rational consumer* can limit the explanatory power of models and can, sometimes, undermine the findings of research.

Alongside these issues, the assumption that all consumers behave rationally, can also be a costly presumption for commercial ventures that are heavily reliant on reliable understanding of human behaviour. Corporations' profitability rests heavily, if not solely, on the ability to produce goods and services that are consumed by individuals. The inability to identify the complex mechanisms that underpin the decision-making process can lead to investments made in inappropriate ventures and products, causing huge losses and undermining the success of businesses.

Therefore, being able to provide any form of insight into the decision-making process can be

valuable. Hence, this project aims to analyze the behaviours in the micro-environment of *The Chase* to illuminate this process.

1.2 Aims

As per the initial submission of the *Description, Objectives, Ethics, Resources* for this project, this report presents our outlined objectives and our success in meeting these goals.

1.2.1 Primary Objectives

I. Create a tool that automatically downloads videos from YouTube, as this is our primary source of data to determine contestants' actions.

II. Use image processing and computer vision to extract decisions of contestants on the game show, *The Chase Australia*.

III. Utilize statistical techniques to implement and conduct preliminary analysis on the decisions of contestants within the specific context of the chosen game show.

1.2.2 Secondary Objectives

I. Utilize further statistical techniques to implement and conduct additional comprehensive analysis on the decisions of contestants within the specific context of the chosen game show.

II. Extend project to a different game show, with a similar concept of decision-making but different competing forces.

1.2.3 Project Success

After careful analysis, we can conclude that our project was successful in meeting *all* the primary objectives, with work being done to meet the secondary objectives.

In order to approach the first primary objective- *creating a tool to automatically download videos from YouTube*, this project utilised *PyTube*- a Python library that facilitates the downloading of YouTube videos from a given uniform-resource-locator (URL). This is discussed in further detail in section 8.1, *Implementation - Downloading Videos*.

Following the obtainment of relevant videos, comprehensive image processing techniques were utilised, in order to achieve the second primary objective - *use computer vision to extract decisions of contestants on the game show, The Chase Australia.*
This project utilises *Open CV2*, another Python library that streamlines the computer vision process. This library was used to extract relevant frames from the videos, such as obtaining the specific frame that showed *all* offers present on screen, the frames that show whether the Chaser and Contestant answer the questions correctly or not, the frames that show which offer was chosen by the Contestant, etc.
Following the extraction of these relevant frames, Open-CV was utilised to extract the *inference* of these frames (for example: deducing that a Contestant has chosen the middle offer when the middle offer turns green), and then written into a CSV file, in order to analyse in the next step.
In this segment, we also utilised the use of the library, PyTesseract- an optical character recognition (OCR) tool, that could extract the text that appeared on screen. This library was used to capture the monetary value of the offers presented to the Contestant.
The implemented Open-CV and PyTesseract functionality is discussed in section 8.2, *Implementation - Computer Vision.*

The approach to the final primary objective and the first secondary objective - *utilise (further) statistical techniques to implement and conduct preliminary analysis on the decisions of contestants* was done through the use of *MatPlotLib, scikit, statsmodels, sklearn* and *Pandas*. *Pandas*, a library created for data analysis and manipulation, was used to convert the initial CSV file into *dataframes*, which streamlined the analysis process significantly. *MatPlotLib* was a graphing library utilised to create intuitive visualisations that provided a strong foundation for the subsequent analysis, as we were able to visually observe the relationships between the decisions made, and potential factors. *scikit, statsmodels* and *sklearn* - three data science analysis libraries were used to conduct regression analyses to estimate the effects of different factors on decision-making. The implemented functionality is discussed in section 9, *Data Analysis.*

2 Context Survey

In this project, the focus is centred around the factors influencing decision-making. As such, this section explores some existing literature that discusses their impacts.

Researching decision-making is the understanding of *how* individuals are able to utilise and prioritise information presented to them. Hence, it is critical to be aware of the way individuals assign weights to the parameters that affect their final conclusions.
This notion is investigated through an experiment presented in the paper, *Judgments of Relative Importance in Decision Making: Global vs Local Interpretations of Subjective Weight* (Goldstein, 1990). This experiment provided metrics about a set of potential apartments to the subjects, and asked them to evaluate each apartment as if *they* were in the market for a specific apartment and then to evaluate each apartment for a friend.
This was an interesting approach to observe the way participants would associate weights to the different criteria corresponding to each house (such as distance from campus, location, space, etc.), and how this weight distribution was updated depending on who they were for. The results of the experiment indicate that individuals' interpretations seemed to be affected most by *relative judgment weights*, i.e, factors such as familiarity with the *content domain* or *target person*. Thus, this communicates that the effects of such intrinsic variables on individuals' weight distributions is an important next step toward further understanding the factors of human decision-making.

Alongside this finding, research that investigates *how* individuals assign weights to different factors, is also very relevant for the scope of this project. Assigning weights is often, not a linear process and is not as straightforward as allocating x points to particular potential outcomes, and can sometimes become quite complex when individuals have to assign weights out of a *budget* of points. (Doyle et al., 1997)
The findings from this paper show that direct ratings and point allocations are not substitutes for each other. In fact, regression analyses conducted by the authors showed two very different weighting profiles emerged when an experiment was conducted where participants were asked to assign weights based on a budgeted point allocation system or a direct rating system. This phenomenon illustrates that this is something to be mindful of, when evaluating the factors that underpin contestants' decisions on *The Chase*. It is important to be mindful of the fact that different contestants may exercise varying degrees of weight profiles when making decisions.

While it is important to evaluate *how* human beings assign weights to decisions, it is also important to consider whether the assignment of these weights actually lead to "good" decisions being made. For the context of this project, a "good" decision is defined as an optimal choice amongst the choice menu.
Summerfield and Tsetsos, 2015) explore this idea by analysing a number of different studies that explore the way participants behave under certain conditions. Scenarios such as asking individuals to choose between two streams of seemingly arbitrary numbers yield the result that individuals tend to have a preference towards the more *variable* of the streams. This is quite an unintuitive result, as one would expect individuals to be more prone to selecting the more predictable of the two streams. However, this is an example of "sub-optimal" decision-making, which provides credence to the notion that the weights placed on various decisions are not always the most desirable. The authors go onto say that individuals suffer from classic economic biases, such as framing and anchoring effects but *also* rely on the adaptation of their neural processing to their local context. Thus, in environments that are unfamiliar or volatile, it can be expected that individuals will make sub-optimal decisions, whereas in stationary and familiar environments, individuals' repeated experiences will enable them to refine their decision making procedure and move towards the "optimal" decision.
Thus, in this project, one can expect there to be irrational behaviour taking place, as it can be assumed that being a contestant on a nationally broadcast television show, with a significant amount of potential winnings, is not an environment that individuals will have faced many times prior. Thus, it can be expected that there are some mappings of neural processing to the new environment that can be observed. As such, it would be interesting to see how the decisions made during this process would differ from the decisions made by the Chasers, as they are much more versed within the environment of the gameshow.

3 Methodology

3.1 Premise

We will begin with a brief overview and description of the premise of *The Chase*. *The Chase Australia* is a quiz show that comprises of three rounds - the *Cash Builder* round, the *Head to Head* round and the *Final Chase* round.

3.1.1 Cash Builder Round

The first round, the *Cash Builder Round* is a relatively straightforward concept. Here, the Contestant has sixty seconds to answer as many questions as possible, earning two thousand dollars ($2,000) for every correct answer.

3.1.2 Head to Head Round

In the second round, the Contestant's goal is to move the money down to "Home" across a 7-step game-board. This can be done by answering the questions correctly, and the number of correct answers required, to secure the cash, is dependent on the offer chosen by the Contestant, i.e, *high, middle* or *low*.
The cumulative amount obtained, as a result of the number of correct answers in the previous round, serves as the *base* offer to the Contestant. If this *base* offer is selected, the Contestant starts three steps from the top, which provides them with a three-step head-start. The Contestant must ***five*** answers correctly, in order to reach *Home*. From this starting point, the Chaser offers the Contestant a *high* offer, and a *low* offer. If the *high* offer is selected, the Contestant only has a two-step head-start and must answer ***six*** questions correctly to reach "Home". Conversely, if the Contestant chooses the *low* offer, the Contestant has a four-step head-start and must only answer ***four*** answers correctly to secure the money.

This is expressed diagrammatically below.

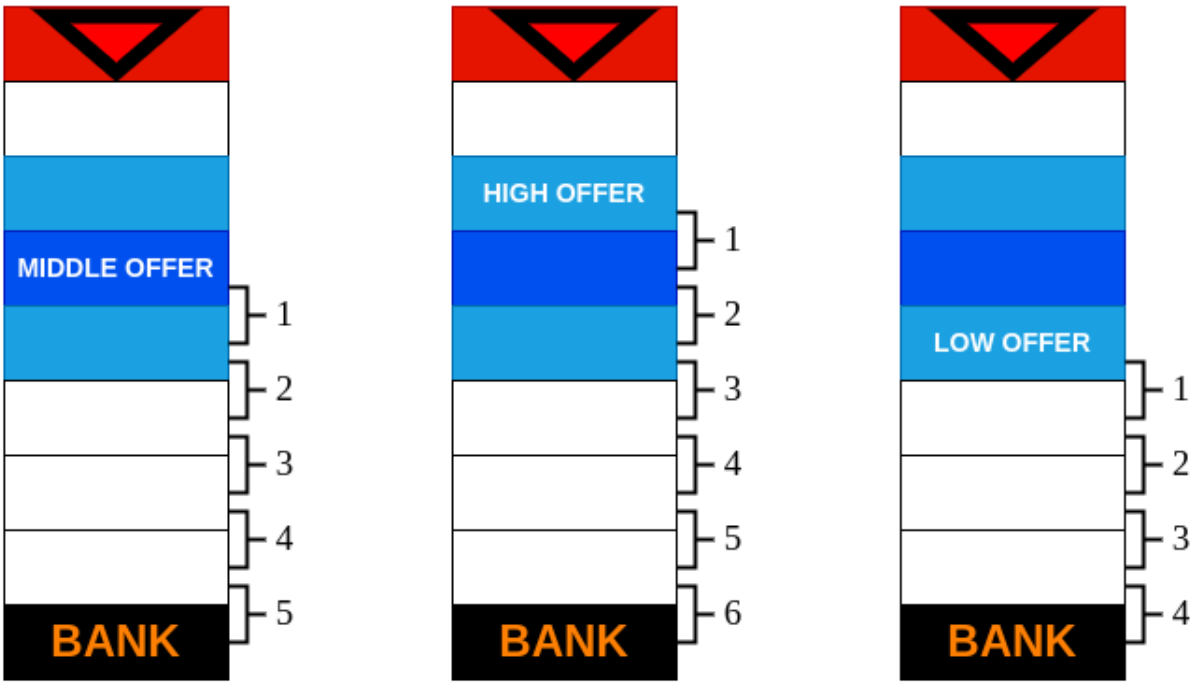

Figure 1: Steps from Home

From the diagram, we can infer that it is significantly more difficult to secure the cash prize when opting for the *high* offer, while the reverse is true for the *low* offer. We can see that every correct answer moves the player *down* one position on the board, and the diagram illustrates the range of correct answers required to win, depending on the chosen offer.
Hence, this project aims to explore the contextual motivation that aim to explain the rationale behind the Contestant's offer selection.

Once the Contestant has selected an offer, the host will ask a series of questions with three answer options. Both, the Contestant and the Chaser must lock in their guesses on the keypads. A correct answer from either party will move their respective position one space down the board. However, a wrong answer will keep the player in place.
If the Contestant can move to the bottom of the board, i.e, *Home*, they can secure the cash prize and add it to the cumulative team prize fund. If the Chaser catches up to the contestant, the Contestant is eliminated and the money is lost.

3.1.3 Final Chase Round

The final round involves the remaining Contestants, i.e, the contestants that made it *Home* in the previous round, working together to beat the Chaser. This is done through choosing one of two unseen problem sets, and aiming to answering as many questions as possible within two minutes. The Contestants are given a head-start of one step per participating Contestant. The Contestants must buzz in, in order to answer, and each correct answer moves the team one step ahead.

Once the Contestants have finished answering the questions, then the Chaser must answer questions from the unused problem set within two minutes. If the Chaser answers incorrectly, the clock is stopped and the Contestants are given the chance to confer and attempt to answer correctly. If the Contestants answer correctly, they push the Chaser back one step.

Through a similar premise, if the Chaser "catches" the Contestants before time runs out, the prize fund is forfeited. If not, then the prize fund is split equally between all remaining Contestants.

3.1.4 The Role of The Chase

Following the understanding of the basic premise of *The Chase* and our rationale, the next steps involved the justification of *why* this particular micro-environment was suitable for our aim - understanding the rationale behind decisions made by humans.

Primarily, *The Chase* features a wide variety of factors that influence the decisions made by the Contestant, when deciding *which* Offer to select. For instance, the order of the Contestant is a prime example of a contributing factor. If the Contestant is the first, they do not have an updated heuristic of the difficulty level of the Chaser, and will have to rely on (assumed) past knowledge. However, if a Contestant is third or fourth, they have an updated and relevant metric of difficulty that will influence their decisions. Hence, one could deduce that the first Contestant might play more cautiously and opt for a lower offer. Alternatively, it can also be deduced that the first Contestant may want to start off strong and gain the respect of their fellow Contestants and thus, opt for a higher offer.

Another interesting metric that affects the decision-making process of the Contestant is the amount of money they may have raised in the *Cash-Raise* round. Naturally, if the Contestant

has answered a number of questions correctly and have a significant amount of capital raised, the safety of the base offer plus its value may convince the Contestant to play it safe, and ensure that this cash prize is added to the prize fund.
However, if the Contestant has already answered a number of questions correctly, they may be feeling extremely confident and may opt for the high offer, as they believe they can take on the Chaser.

Additionally, a well-performing Contestant may be a threat to the Chaser, whose main incentive is to eliminate the Contestant. As such, they may try to tempt the Contestant with an absurdly high offer, in an attempt to maximise the Contestant's chances of being eliminated. This particular motivation may be encapsulated in the deviation of the Chaser's offers when compared other offers made.
Alternatively, the Chaser may not want to tempt fate if a Contestant has demonstrable skills, and as such, may *not* offer a scandalous amount, lest the Contestant accepts this challenge and wins.

Hence, one can see that there are a number of competing factors that influence the decisions made by the Contestants and the Chaser. It is precisely these influences that this project aims to analyze and discuss, with the goal of identifying the dominant rationale.

4 Requirements Specification

The requirements for this project were developed at the start of the project, and have been specified in the Description, Objectives, Ethics and Resources (DOER) Report.

These objectives are specified as follows:

4.1 Primary Functionality

4.1.1 Primary Objectives

I. Create a tool that automatically downloads videos from YouTube, as this is our primary source of data to determine contestants' actions.

II. Use image processing and computer vision to extract decisions of contestants on the game show, *The Chase Australia*.

III. Utilize statistical techniques to implement and conduct preliminary analysis on the decisions of contestants within the specific context of the chosen game show.

The aforementioned objectives are the core functionality of this project, as this project must be able to parse a text file of video URLs and download them, because these videos will serve as our entire data set for this project.
Once this can be successfully completed, the software must be able to extract appropriate information from the data set acquired from Objective I. Finally, the software must be able to perform statistical analysis on the extracted information from Objective II.

These core objectives are listed in the order of priority, and illustrate the different components that must be successfully completed over the duration of this project.

The following section illuminate further details about these primary objectives, and refines these into a set of tangible of requirements that can be used as a foundation for the development process of this project.

4.1.2 Primary Functional Requirements

I. The software shall be able to download videos to the hard drive, when provided with a list of URLs.

II. The software shall be able to read a video file from the hard drive.

III. The software shall be able to extract specific frames through analysing the pixel density at specific positions within a frame.

IV. The software shall be able to draw appropriate conclusions about the specified frame being analysed through logic checks, such as identifying whether it is a Question Frame, an Offer Frame, a Choice Frame, etc.

V. The software shall be able to extract key information from identified frames, such as identifying whether the Chaser/Contestant gets the answer correct, the choice they make, the number of answers correctly answered in the Cash-Raise round, etc.

VI. The software shall be able to write the extracted key information to a CSV-file.

VII. The software shall be able to process and apply appropriate filters to the Offer Frame (shown below), in order to prepare the image for optical character recognition.

Figure 2: Offer Frame

VIII. The software shall be able to use Optical Character Recognition software, i.e, *PyTesseract* to identify the monetary value of the offers presented to the Contestant, by the Chaser.

IX. The software shall be able to perform checks that the optically recognised numerals are expected values.

X. The software shall be able to attempt to correctly read an initially misidentified frame, through the use of additional or alternative pre-processing techniques.

XI. The software shall be able to write the identified monetary value of the offers to a CSV-file.

XII. The software shall be able to read from the CSV-file, containing the extracted information and monetary values mentioned above.

XIII. The software shall be able to create a data-frame from the information contained within the CSV-file.

XIV. The software shall be able to analyse the information contained within the data-frame, in order to conduct statistical analysis.

4.2 Secondary Functionality

4.2.1 Secondary Objectives

I. Utilize further statistical techniques to implement and conduct additional comprehensive analysis on the decisions of contestants within the specific context of the chosen game show.

II. Extend project to a different game show, with a similar concept of decision-making but different competing forces.

4.2.2 Secondary Functional Requirements

The first secondary objective is an extension of the last primary objective, and as such, the first fourteen functional requirements for this objective remain the same as those outlined above.

The second secondary objective is an extension of the full project that has been described above. Thus, the functional requirements for this aim is the same as the first six functional requirements outlined above. The only deviation is that the program's parameters for identifying a *relevant frame*, as per the expected pixel composition, will change.
After the first six functional requirements, the requirements will evolve to suit the game-show being analysed.

For example, analysing the game show *Deal or No Deal* may have functional requirements that allow the software to analyse the percentage of times a Contestant will choose to swap their box at the end of a game, or accept the Banker's Offer.
The optical character recognition tool would be instrumental when analysing this particular game-show, as it would be essential to identify the amount contained in each opened box, as this would influence the Contestant's confidence of whether they had a high offer in their box or not, thus influencing their likelihood of accepting the Banker's Offer.

Hence, the program could serve as a framework that can be tweaked, in order to meet the demands of analysing different game shows, as the foundations of pixel density analysis, logical inference, optical character recognition and statistical reasoning, have already been laid.

5 Software Engineering Process

5.1 Development Process

During the undertaking of this project, I followed an *Agile* development process the most closely. Due to being unfamiliar with image processing and computer vision, which would constitute a large portion of this assignment, it was essential to adopt an approach that included the scope for continuous testing and assessment of the project, in order to ensure that the software was advancing at an appropriate pace.

The main developmental process of this software project followed the key principles of breaking down each task into smaller components, which could be completed fortnightly. These developmental cycles can be compared to the *sprint* cycles that are outlined within the *Agile* development process.
The objectives that had been specified in the DOER (Description, Objectives, Ethics & Resources) document at the start of the project, were extremely useful during this process, as these goals served as a *Product Backlog*, which could be referred to, during the development process. As these objectives were listed in order of priority, it was very simple to compose a list of functional requirements from each aim (as outlined in *Section 3 - Requirements Specification*), that acted as the *Sprint Backlog* during the specific *sprint* cycle.

Testing was also an integral part of these sprint cycles. While this aspect of the iterative approach was not explicitly specified in the DOER, this was essential component of this project, as it was critical to ensure that the implemented image processing techniques worked appropriately, and were able to extract the relevant information and conclusions.
The most important aspect of this project was to ensure that *all* the relevant information was captured during an iteration of the program. As each episode consisted of four Contestants, the program needed to capture four Offer frames that were distinct from each other, four Offer decisions and somewhere between sixteen to n Question frames.
As such, it was crucial to create a robust testing harness that was able to ensure that the program was performing as expected. Hence, extending the testing suite to accommodate the inclusion of new features was also a critical part of the iterative development methodology. In order to approach the testing facet of this project, I observed 10% of the data-set (approximately, 5 videos) and created a set of unit tests that checked whether the software was able to extract the expected information.
As the project evolved, I extended the functionality of the Test Class, in order to ensure that

the added components were behaving appropriately. This process is discussed further in the *Testing* section.

This approach was supplemented by meetings with my supervisor, wherein we conducted meetings on an *ad-hoc* basis. During these meetings, I would inform my supervisor of the work that had been completed since our last meeting, and how this work contributed to the objectives that we had specified, at the start of the year. I also consulted with my supervisor about any problems that had arisen during the course of these development cycles and how best to approach them.

While this development process was similar to an *Agile* development process, there were certain aspects of this process that did not represent the quintessential expectations of *Agile*. Primarily, as this was an individual project, the team-based approach of this process was not fully capitalised on. Properties such as the *daily scrum* were not included in the undertaking of this project.
Furthermore, while this was an independent project, it was not a full-time commitment. This venture was undertaken alongside six other modules, and as such, all the virtues of an *Agile* project could not be preserved, as there were other coursework deadlines or exam periods that needed to be factored into the development process. Hence, the sprint backlog could not always be completed within a pre-specified amount of time. Because of this, the sprint cycles would often vary in lengths, depending on the complexity of the work or other upcoming academic commitments.

However, despite these constraints, the *Agile* method was still the best approach for this Senior Honours project, as the iterative development process enabled Dr. Arandjelović and I to adjust our time-frames according to the upcoming constraints. For example, during extremely strenuous periods, such as the exam period of Semester 1, the project was scaled back to allow work on smaller components (such as minor refactoring or improvements to the program's accuracy), whereas periods such as the Christmas break, or the first three weeks of Semester 2 were utilised to complete major changes, such as major refactoring or major functionality additions.
Hence, while there were no changes to the major objectives of the project, the *Agile* method still enabled us to be flexible on when each item of the product backlog was worked on.

An example of an amended sprint cycle is one that begun after Independent Learning Week

(ILW) to Week 8 of Semester 1, which consisted of working on the image processing aspect of the program- specifically, ensuring that the program was able to recognise the four distinct Offer screens for the four Contestants. This was initially considered to be a relatively simple task, but quickly revealed itself to be quite complex, as this frame would appear multiple times throughout the episode, and the program would constantly reset every time this frame appeared on-screen. This was because the program was configured to recognise the Offer screen as **the start of a new Contestant and reset**, and as such, it was imperative that the program only picked up the first time the Offer frame appeared for a Contestant. Hence, this task ended up taking longer than expected, which made the rest of the development cycle slightly slower.

Thus, this is a prime example of why the *Agile* methodology was useful in this project, as it allowed us to re-engineer the requirements for this functionality, try an alternative approach and adjust the sprint backlog for the subsequent sprints.

5.2 Tools & Technologies

A multitude of resources and technologies were utilised during this project. This section explores the different tools used, and offers a justification for the utility of these instruments in the project.

5.2.1 Python

The entirety of this project has been written in Python. Python was used for this project as it is ideal for statistical analysis and data manipulation, due to the presence of libraries such as *Matplotlib, Pandas* and *NumPy*.
Python also supports the use of the *Open-CV* library, which is used for image processing and computer vision in this project. This made this aspect of the project much more streamlined.

5.2.2 PyTube

PyTube is a Python library that has been used to download videos from YouTube. It uses an API to download YouTube videos, and allowed us to specify the desired resolution type. This streamlined the data-collection process, as we were able to write a simple function that downloaded a list of videos, in the highest resolution available, once a text-file of video URLs were supplied. 2

5.2.3 Open-CV

Open-CV is an extremely popular library that is used extensively for computer vision. We utilised this library due to the extensive documentation available, as well as the wide array of image-processing tools embedded within Open-CV. This library enabled us to extract key information from the initial data-set and pre-process the Offer frames, in preparation to "read" the text on screen. 3

5.2.4 PyTesseract

PyTesseract is an optical character recognition tool for Python, that was used to identify the monetary value of the Offers presented to the Contestant. 4

5.2.5 Pandas

Pandas is the data analysis and data manipulation tool that was utilised for the statistical inference aspect of this project. The information extracted from the computer vision stage was read into a data frame, supported by Pandas, allowing for easier analysis. Pandas was also used to perform regressions, in order to find the constituent factors of individuals' decisions. 5

5.2.6 Matplotlib

Matplotlib is a Python library that facilitates the creation of interactive or static data visualisations. This has been a critical tool to explore any relationships that may exist amongst factors, such as the relationship between the order of the Contestant and their choice of Offer. Matplotlib also helps plot the range of Offers presented, which allows us to make inferences about the types of Offers that can be expected.6

5.2.7 GIT

GIT was used for version control during this project, by ensuring that all code and local changes were appropriately backed up. This would allow for instant recovery, in the case of local hard disk failure. 7

6 Ethics

There were no ethical considerations for this project, as all the data was publicly available on video streaming websites.

The downloading of the videos, in order to construct a data-set, is compliant with video streaming websites' terms and conditions, and further supported by the outcome of *Sony Corp of America v. Universal City Studios, Inc. (1984)* case [1] .

This project contains no identifiable information of individuals, and only focuses on the accuracy of contestants' answers and their decisions of which Offer to pick.

7 Design

This section will discuss the design decisions and strategies employed, in order to complete the requirements outlined in *Section 3- Requirements Specification*.

7.1 Downloading Videos

The first stage of this software project, was to design a tool that would enable the program to download videos from YouTube.

The main aim of this functionality was to make data collection as streamlined as possible. As we were planning on collecting numerous videos, it would be time-consuming and inefficient to have to download videos manually, through third-party software.

Hence, this process was made more efficient, by composing a list of URLs into a file, and providing this to the program, in the form of a command-line argument.
For the purposes of this project, we wanted our data-set to comprise of videos of *The Chase*, in the *highest resolution possible*. This was because we wanted our data to be high quality, in order to ensure that our consequent computer vision and optical character recognition stages would behave appropriately.

As we only planned on running this download function once, the optimization concerns surrounding the download and analysis of multiple 720p videos was not a critical consideration. The purpose of these high-quality videos was to extract the relevant information from these videos *once*.

7.2 Computer Vision

The second stage of this software project, was to use image processing techniques on the downloaded videos from the previous stage, and extract *all* key information. This was done through the use of Open-CV2, a Python computer vision library.

The following flowchart describes a rough outline of the steps that the software needs to take, in order to obtain the relevant information.

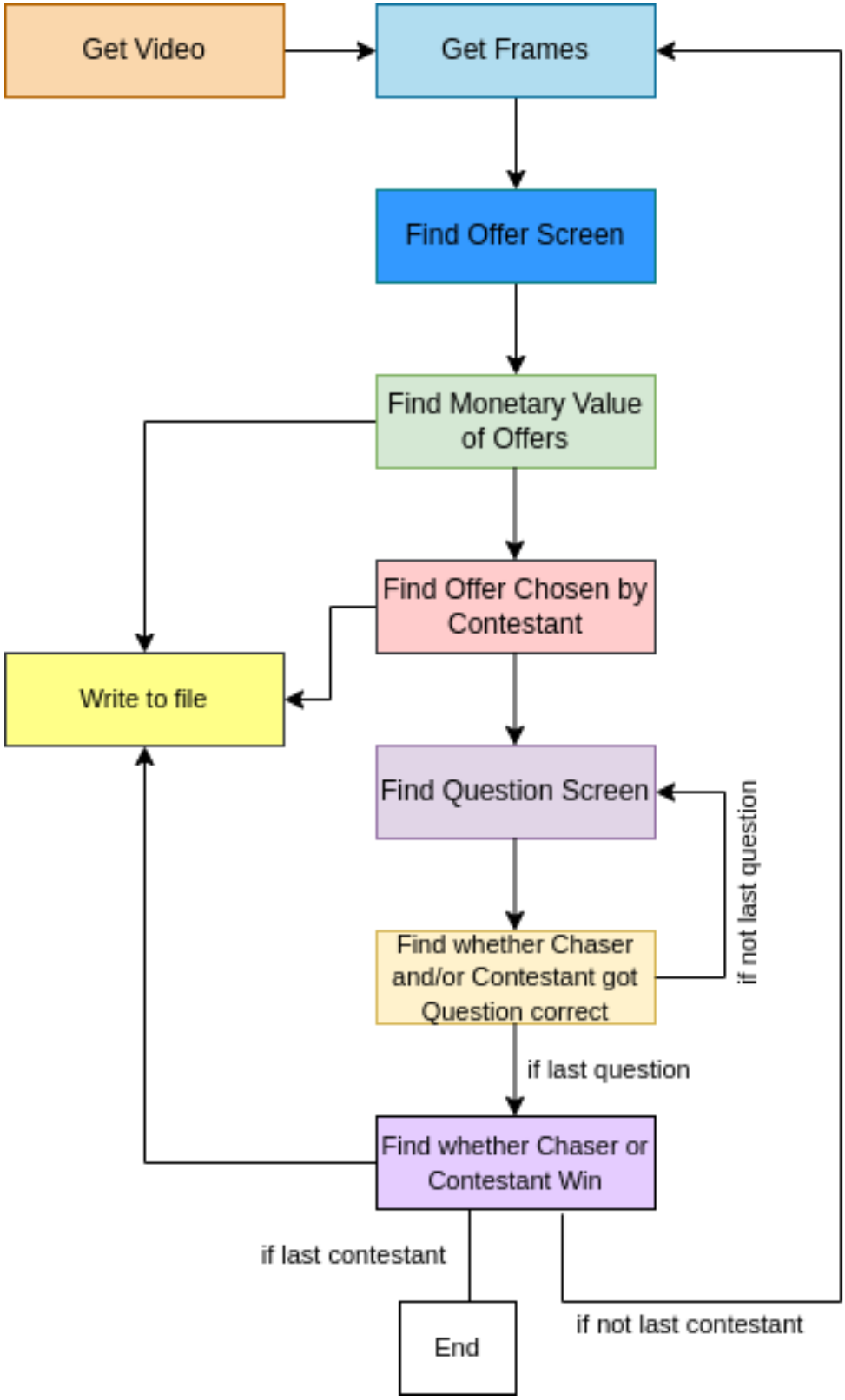

Figure 3: Flowchart of Program Steps

As we can see from the diagram, the program has a finite number of steps that must be taken, once a video is supplied to it.
Primarily, the software must be able to read in a video, and parse through the pixel composition of the frames, in order to determine whether a frame is relevant or not.
The first frame that is deemed 'relevant', is the Offer screen, that depicts the three offers presented to the Contestant. An example Offer frame, is shown below.

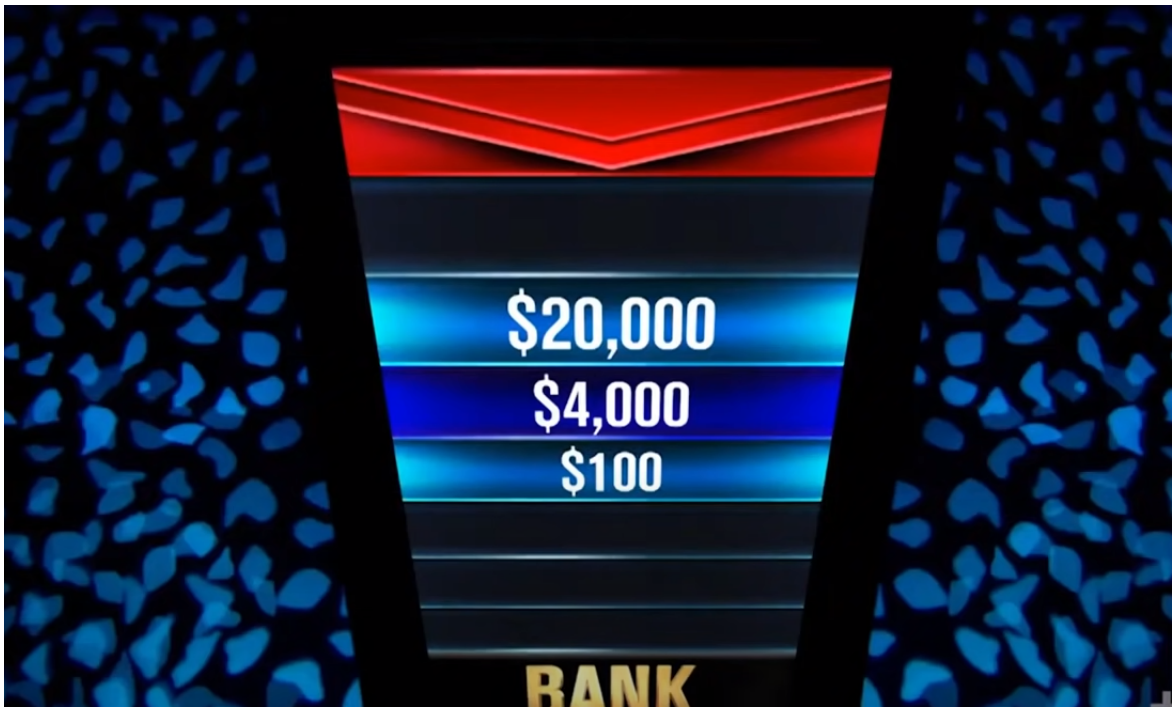

Figure 4: Offer Frame

From this frame, we expect the program to be able to 'read' the text on the screen ($20,000, $4,000 and $100), using an optical character recognition tool. We want this information to be written to file.
If this is the first Offer frame that has been seen by the program (for this particular video), the program should assign '1', to indicate that it is currently the first contestant's turn.

Following this, we expect the program to be able to find the frame that depicts which offer has been chosen. An example Decision frame, is shown below.

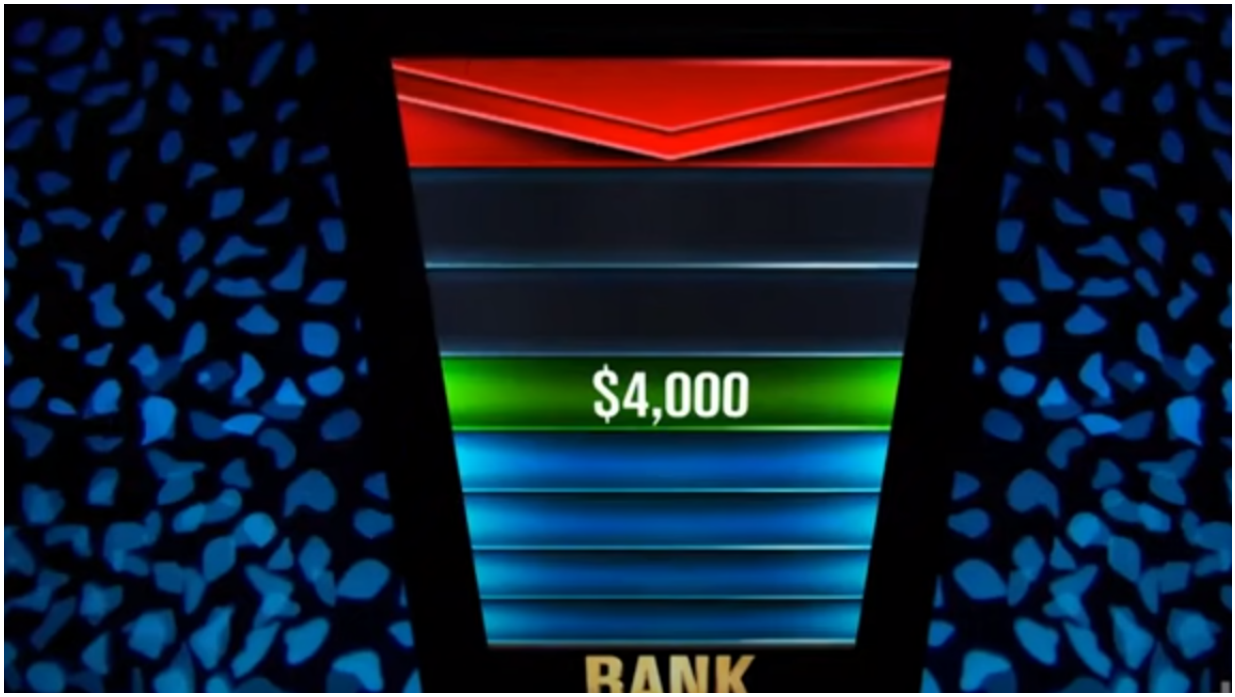

Figure 5: Decision Frame

From this frame, we want the software to be able to recognise that the Middle offer has been selected. This can be done through identifying that the pixel density in this area is predominantly green, hence, the Middle offer has been chosen.
Next, we want to write the Contestant's Decision to file.

Once this has been done, we want the program to be able to deduce the number of steps that are required to secure the cash prize and reach *Home*. The diagram in *Section 1- Introduction* of this report, indicates that it is 6 steps to *Home* when the High Offer is selected, 5 steps to *Home* when the Middle Offer is selected, and 4 steps to *Home* when the Low Offer is selected.

Following this inference, we move onto the *Question* segment of the Contestant. Specifically, we want to be able to identify the Question frames. This enables us to have a clear understanding of when we are within the parameters of a specific question, and can identify whether the Chaser/Contestant have got that specific Question correct.
An example of a Question frame is shown below.

Figure 6: Question Frame

Being able to map a Response frame to its relevant Question frame is important, as this allow the program to demarcate between Questions, and not get confused about which Response frames correspond to which Question frames. This is because Response frames remain on screen for multiple seconds, and the only way for the program to identify when the show has moved on from a specific question, is by seeing the next Question frame.
An example of the relevant frame sequence, in this segment, can be seen in the diagram below.

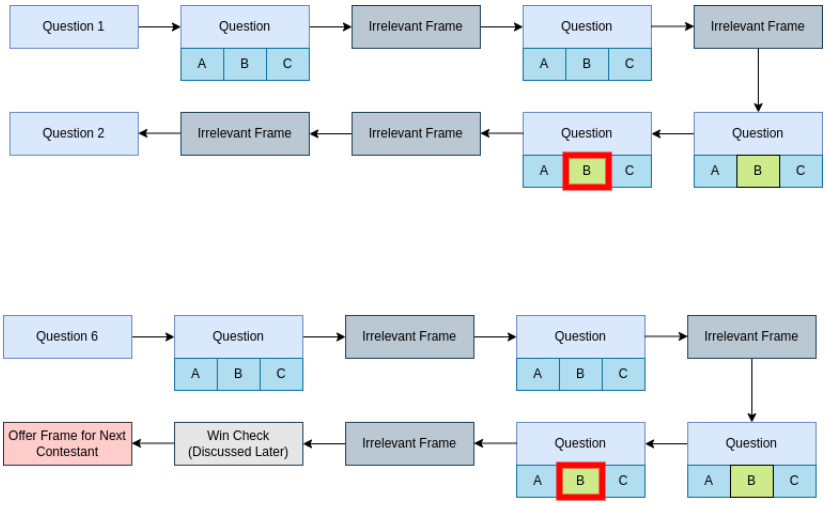

Figure 7: Relevant Question Frame Analysis

From this diagram, we can see that a Question frame is a distinct feature that signifies either the beginning and/or end of a Response segment. This allows the program to only look for Response frames within the boundaries of a Question - Question frame pair, or a Question - Offer frame pair. This ensures that the Response frame obtained, is mapped to the correct Question.

An example of a Response frame is shown below. This constitutes as a Response frame as it contains the information of *both*, the Chaser and Contestant's position on the board, i.e, whether they will proceed to move down the board because they got an answer correct, (as explained in *Section 1 - Introduction*), or stay where they are because they got an answer wrong.

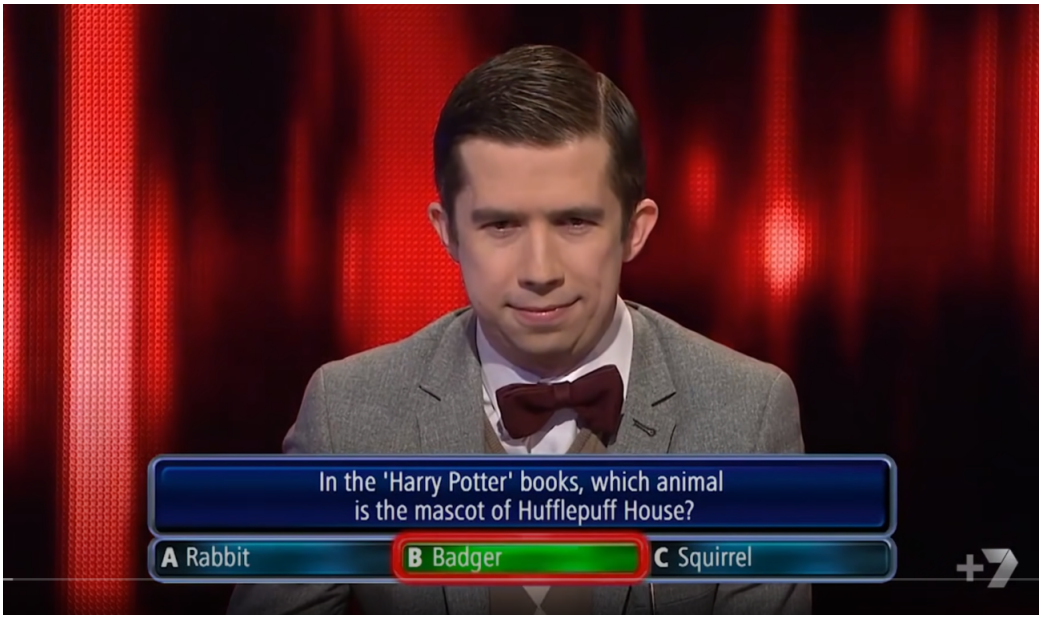

Figure 8: Response Frame

The Question segment lasts until the Chaser or Contestant win. The program is able to identify this exit condition by recognizing the Offer frame reappearing. The program then moves back to the top of the flowchart depicted earlier in this section, and restarts the process.

The win condition of whether the Chaser or Contestant wins, is done through checking the number of questions that the Chaser and Contestant have gotten correct.
This is done through the program identifying the number of questions required to get *Home*, depending on the Offer chosen. Once this information has been extracted, the program counts the number of questions that the Chaser/Contestant have gotten correct. If the

Contestant reaches *Home*, i.e, they reach *0*, they have won. If the Chaser catches up to the Contestant, i.e, they are the same amount of distance away from *Home*, it means that the Chaser has won.

It is important to refer back to the flowchart at this point, and ensure that all the relevant information is being written to file. The key information extracted from the files are as follows:

- Episode
- Order of Contestant
- Monetary Values of Offers
- Option Chosen
- Outcome of Contestant (Win or Loss)

It is these pieces of information that will populate our data-frames in the next step, allowing us to conduct data analysis.

As noted in Figure 3 and its subsequent discussion, it is evident that there were multiple components in the image analysis aspect of this project. Each component is responsible for executing a specific type of functionality, such as finding the Offer screen (as seen in Figure 4) or finding a Question Screen (as seen in Figure 6), etc.
Thus, the most intuitive approach to this problem, is to develop the project in a similar modular manner, and break down the required functionality into constituent components. These components would then be connected with each other, after its successful development.
This perspective would allow for efficient testing, as it would facilitate the creation of a full testing harness after the origination of each component. This would provide assurance that each individual component within the system was functioning correctly, including edge cases, before being connected to the rest of the program. Any issues that arose as a result of this integration would allow for immediate identification of the problem areas, which would make these issues easier to remedy.
Overall, the adoption of this component-based architecture would make the development process much smoother, as each component was developed in an insular manner, and testing each component (in isolation and in conjunction with the rest of the program), made it much easier to evaluate and assess the unit's performance.

7.3 Data Analysis

In the final stage of this project, the extracted information from the previous stage served as the data-set for the statistical analysis.

As the extracted information was in the form of a CSV file, the initial approach to analysing this data was through simply reading in each line as an array, and accessing each piece of information by its associated index. While this is a viable approach, it is more cumbersome to analyse the extracted information in this manner, as it would require the indices of key pieces of information to be hard-coded into the program, constant looping of each array, and other such limitations.
In lieu of this approach, the approach was amended, in order to adopt the use of *data-frames*. A data-frames is a two-dimensional data-structure, which organises information into rows and columns, and behaves similarly to a spreadsheet. Data-frames also have a significant amount of in-built functionality, such as indexing data by column name (instead of hard-coding an index and looping through each array), adding, deleting or renaming columns, checking for and dropping missing values, etc. This makes it significantly easier to manipulate data due to these advantages.

Following the adoption of data-frames as the main data-structure containing all relevant pieces of information, it was extremely easy to extrapolate the necessary information in order to create data visualisations that highlighted the various relationships between data. These visualisations are presented in *Section 9.2 - Data Visualizations*, illustrating the various relationships and dynamics between metrics. This makes it much simpler to obtain an intuitive understanding about the factors being considered, while facilitating the identification of any patterns that may exist.
Alongside this, the extrapolated data was also utilised to conduct various regression analyses to obtain a deeper understanding of the constituent factors that influence Contestants' decision-making process. These regression analyses are discussed in greater detail in *Section 9.3 - Methods and Findings*.

The main aim of conducting these regressions was to find the weights, or the extent of the effects of certain factors, on certain dependent metrics. Specifically, the objective was to find the influence of factors such as the *order* of the Contestant, the amount of money in the prize fund, the number of Contestants who had made it through to the final round and the Offers presented, on the **Contestant's decision of whether they would pick the High, Middle or**

Low Offer.
This analysis, therefore, provides some insight into the influences behind individuals' decision-making processes, when evaluated in a controlled environment with a limited number of factors.
Thus, a wide variety of statistical techniques and machine learning models were used, in order to evaluate and assess the significance of the different features incorporated within the feature matrix, as calculated by the different approaches.

7.3.1 Linear Regression

Linear regression is a fundamental statistical technique that enables data analysts to observe the relationship between a dependent variable and one or more explanatory or independent variables. A linear regression model was the first approach that was undertaken in this project, as it is the most straightforward strategy to obtain some understanding about the basic relationships between variables.

An example of a basic linear regression equation is presented below.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{1}$$

where,
Y = dependant variable
β_0 = intercept
β_1, β_2 = slope coefficients
X_1, X_2 = explanatory variables
ϵ = error term

For the purposes of this project, an *ordinary least squares (OLS)* estimation was used in order to obtain the unknown parameters of the model, i.e, the coefficients of the explanatory variables of the model.

The OLS technique is able to obtain these coefficients by minimising the *sum of squared residuals*, i.e, by minimising the difference between the *predicted* values and the *actual variables* of the dependent variable (Y). This is done for every observation in the sample, and

these differences are then squared to produce the *best line*, i.e, the line with the lowest sum of squared errors (SSE).

The reason that the OLS estimation technique was used for this project, was because it is the most popular estimation technique due to its property of being BLUE (best linear unbiased estimator). This basically indicates that OLS estimators are linear, unbiased and have the least variance amongst all linear and unbiased estimators. Hence, this made OLS the most fitting technique to be utilised.

7.3.2 Logistic Regression

Following the linear regression technique, a logistic regression was employed, in order to further understand the relationship between the dependant variable and its explanatory variables. This method enables the program to predict a dependent data variable, based on prior analysis and observations of one or more explanatory variables. Hence, the line of best fit can be obtained by optimising the parameters of the function, in order to minimise the cost function.

A logistic regression is defined by its cost function. A cost function is the loss function plus a regularisation term. Typically, a loss function is used to evaluate how well the model can predict the behaviour of the data, which is canonically, the sum of squared errors. The loss function punishes the model for wrong predictions, in order to improve the model to fit the data pattern and promote improved predictive performance capabilities. The regularisation term is used to generalise the line of best fit, and avoid over-fitting.

The L2 regularisation prevents the coefficients of each explanatory variable being non-uniform. This is done through adding the sum of the squared coefficients to the loss function, as a penalty term that needs to be minimized. Thus, this was used for this project, as it shrinks the weights of the coefficients evenly, and as such, is useful when there may be co-dependent features, as in our case.

7.3.3 Random Forest Regression

The next regression technique utilised is a Random Forest Regression- a machine learning model. This is a supervised learning algorithm, that uses an *ensemble learning* method for

regression - a technique that combines predictions from multiple machine learning algorithms.
This allows random forest regression to produce a more accurate prediction than a single model. The predictor is created through constructing several decision trees during the training period, and presenting the mean as the prediction of all the trees. Within a random forest regression, there is no interaction between the constituent trees.

Hence, random forest regression was employed for the purposes of this project, as it reduces the issues of over-fitting that other models face, due to its averaging of solutions produced by each decision tree. Within Python, using the *sklearn* module, the importance of individual features that were utilised to train the random forest regressor can be obtained. As such, it would be interesting to view the results obtained from this technique.

7.3.4 Support Vector Machine (SVM) Regression

The final technique adopted for this project was the utility of a Support Vector Machine (SVM) Regression. Like its predecessor, SVM regression is also a supervised learning technique within the realm of machine learning.
An SVM works by presenting the hyper-plane that best separates different data points. A hyper-plane, put simply, is a line that divides the classes of data, when considered in 2D-space. This line is known as the *decision boundary*, i.e, the demarcation to classify each point, depending on which side of the line it falls on.
The data points that are closer to the hyperplane are the main influence over the position and orientation of this line, and are commonly known as the support vectors. These are selected to maximise the margin of the classifier, which is the main goal of this regression technique.

7.3.5 Polynomial Features

In order to further maximise the benefits that could be obtained from the SVM model, the use of *polynomial features* were also investigated. This allows the model to consider the given features of input samples, but also the combinations of the input matrix, commonly known as the *interaction features*.

The generation of these new features enables the program to survey new combinations and relationships between explanatory variables and the dependent variable, that might be useful to machine learning models, specifically, the SVM model, as it allows the model to map linear

variables into a non-linear feature space.

7.3.6 Data Visualization

The final form of data analyses conducted was the *data visualisation* facet of this project. This stage allows for easier comprehension of relationships between features. This is due to the interactions being modelled graphically, using *matplotlib*. The insights obtained from these data visualisations underpin the expected results from the above statistical analysis techniques.

8 Implementation

8.1 Downloading Videos

This would involve the composition of a file that contained a list of URLs of videos that would be downloaded, with each new line corresponding to a single URL, i.e, separated by a newline separator. The path to this text-file would then be supplied to the program as a command-line argument, which would allow the program to read in the contents of the file, line by line.

The composition of these URLs was quite simple, as we were fortunate to find the vast majority of our data set in a playlist on YouTube. As this part of a playlist, the body of the URL for each video remained the same across the playlist. The only difference between the videos, was a changing suffix at the end of the URL, which was a number that began at 1 and went up to 44, to signify the 43 videos included in the playlist. Thus, it was easy to generate these URLs through a simple *for loop*.

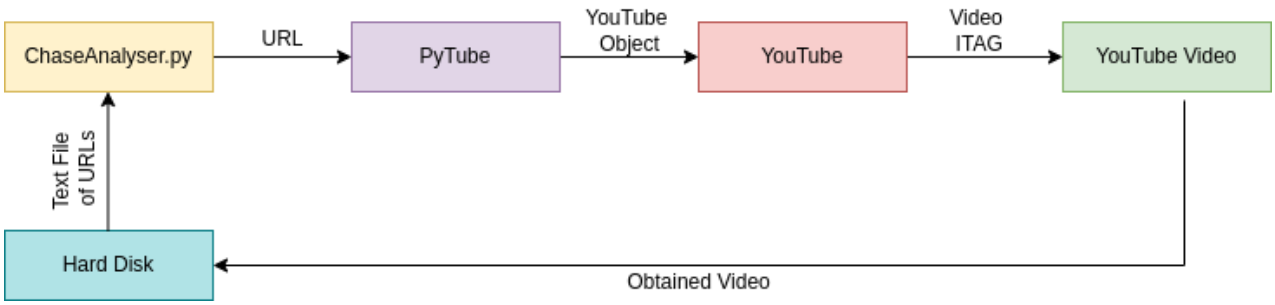

Figure 9: Flowchart of Video Downloading Process

8.2 Computer Vision

The first stage of this process was to get the frames from each video, in order to analyse the pixel composition of each frame. The frame rate of the video was 30 FPS (frames per second), and as each video was approximately, 42 minutes in length, this would indicate that the program needed to check 75,600 frames to find all the relevant frames in a video.

This vast amount of checking appeared to be excessive, hence, the program was set to only analyse every third frame. This cut down the analysis time by a significant amount, while still maintaining the integrity of the project, as the program was still able to retrieve all key pieces of information.

It may seem odd that optimisation concerns were sacrificed in the first stage of the process, in order to download high-resolution videos, but *some* optimisation concerns were integrated in the second stage of the process. This was done because the computer vision stage involved a significant amount of testing of frames. Hence, reducing the number of frames to be analysed significantly sped up the testing process, which made it much faster to identify issues and progress.

Following this, the pixel composition analysis stage commenced. As mentioned in *Section 7.2 - Computer Vision*, a component-based approach was undertaken for this section.

The first step was to read in each video, on a frame-by-frame basis. This allowed the components within the frame to evaluate each frame in isolation. Next, the components were developed. These are described below.

8.2.1 Answer Frame Component

The first component was created to extract the frames that depicted the Chaser's and the Contestant's responses to a question, as well as the correct answer- an *Answer* frame. An example Answer frame is shown below.

Figure 10: Answer Frame

From the example, it is evident that there are a number of unique colours that can be used to

evaluate whether a particular frame contains relevant information. These colours are:

- Dark Blue: Answer box that was *not selected* and is *not correct*.
- Red: Answer box chosen by the Chaser.
- Green: Correct Answer box.
- Light Blue: Answer box chosen by the Contestant **if** they have answered incorrectly. Not always present.

Hence, in order to flag a specific frame as an Answer frame, the program would search for a combination of the above colours, i.e, whether there was some pattern of red/dark blue/light blue/green. As noted above, if the Contestant was able to answer the question correctly, there would be no light blue Answer box. Hence, this would be replaced with a dark blue Answer box. Thus, the identification of these four colours made it very simple to configure a colour palette that comprised of the RGB (Red, Green, Blue) values of the four colours, enabling the program to "grab" the correct frame.

Once this pattern had been identified, it was relatively straightforward to implement this functionality.
First, it was important to identify the different pixel ranges that corresponded to the colours within each Answer box. As seen from the image, it is evident that there is some shading in each Answer box, i.e, it is not one solid colour. Thus, in lieu of hard-coding one specific value, the program accepted a *range* of RGB values, that would be used to identify the colours within a specific location. The correct RGB values were obtained through the use of an online pixel-examination tool, *Pixspy*, which made it easy to obtain an estimation of the range of pixel values that represented each colour. It was important to maintain that there was as little overlap between pixels as possible, especially the dark-blue and light-blue colours. The reason for this is explained in the next section.

Once the range of pixel values had been obtained, a *colour_checker* function was created, that would do a simple check of whether a pixel's RGB composition matched any of the identified colours. This functionality was then tested, by passing in a range of Answer frames to the program, and checking whether it was able to identify the appropriate colour. This was done to identify whether there were any RGB values that fell outside of the given range, and if so, this was amended.

Following this, the next piece of functionality was designed. This was the *logic()* function, which took in a frame as a parameter, and checked the RGB values of pixels at a specific, hard-coded value. This was done as the videos were all obtained from the same source. Thus, the placement of pixels of a specific colour palette was always the same. Hence, it was possible to set static coordinates, i.e, x and y coordinates that would check the pixel composition at a particular location. If this was not the case and it was not guaranteed that this information would always be at the same location, the program was able of checking the bottom quarter of the frame.
However, for the purposes of this project, the specific y-coordinate and x-ranges were given, instead of checking the full bottom quarter of the frame. Thus, this enabled the software to run much faster.
Next, this function checked the RGB values of the pixels within this specific range, by calling the *colour_checker* function. If any pixel within this range matched any of the RGB colour values, the function would return the matched colour, which would then be appended to an array.
It is important to note that *if a pixel did **not** match the RGB composition of any colour*, it would simply return BLACK. This would not be added to the array. Additionally, if the function matched the RGB composition of the same colour consecutively, the repeated colours would *not* be added to the array. This is because an Answer box is multiple pixels in length, and as such, it can be expected that most, if not all, of those pixels will return the same dark blue/light blue/green colour. Hence, just identifying the colour of an Answer box once is enough, as this will provide the program with sufficient data to analyse whether the Contestant or Chaser got the answer correct.
This continues until the program reaches the end of the specific x-coordinate range to check.

The logical flow of the described functionality is expressed below:

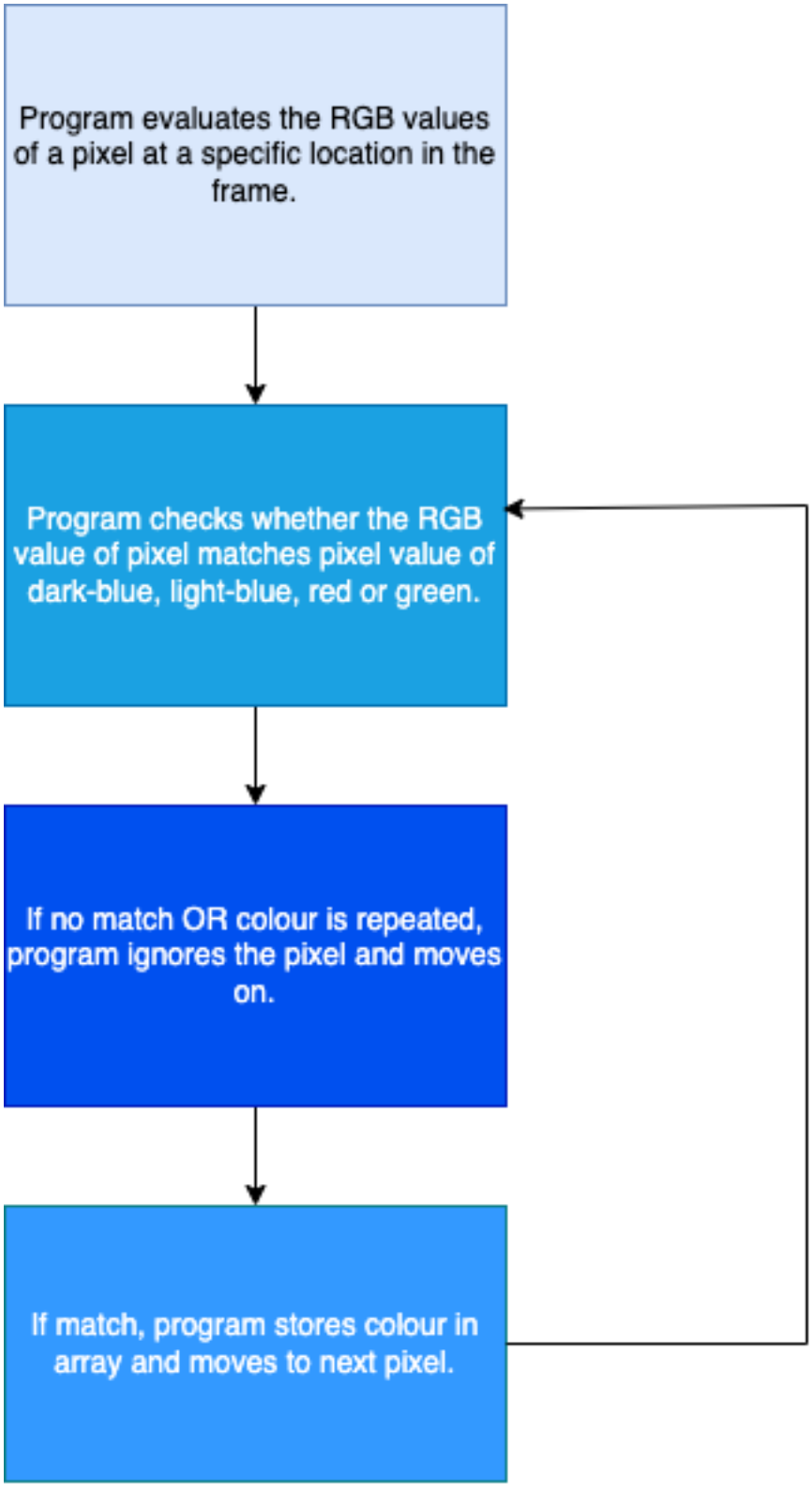

Figure 11: Flowchart of colour_checker Function

Thus, once the supplied frame has provided the relevant information, i.e, the appropriate placement, the program moves onto analysing this data.
As mentioned earlier, this information is presented in the form of an array. Specifically, every time a new colour (Light Blue, Red, Green, Dark Blue) is identified, it is appended to an array, which is then passed into the *inference* function.
This function takes in the array, and analyses the placement of the different colours, in order to infer whether the Chaser and/or Contestant got the question correct. This is expressed diagrammatically below:

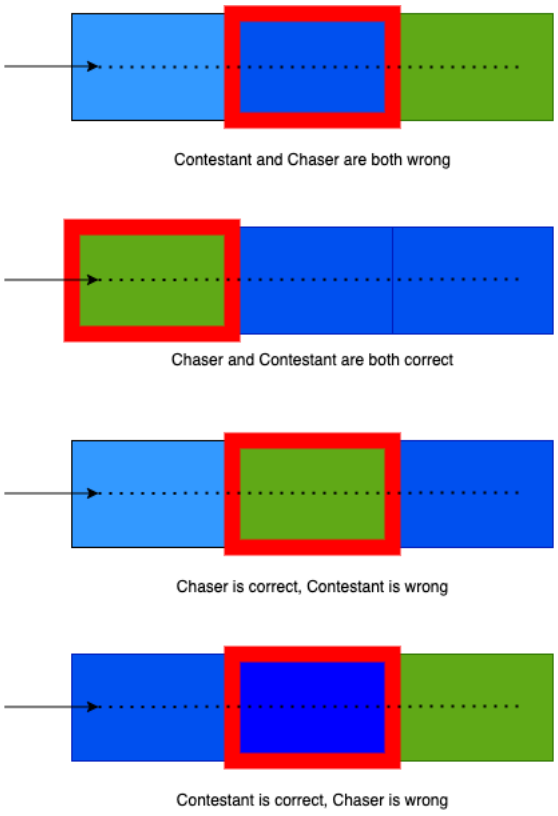

Figure 12: Colour Placement Analysis

Firstly, it is important to note that the starting x-coordinate of the Answer box analysis is slightly more inward than anticipated, i.e, the program does not start analysing pixels from the very first pixel that represents the box. The final x-coordinate that is checked, is also not the last pixel of the box.
This is a design decision that was made during the implementation phase, as it made it easier to infer whether the Chaser was correct or not. This is explained in detail below.

From the above diagram, it is evident that the placement of the specific colours enable the program to quickly infer whether the Chaser and the Contestant were able to answer correctly. For example, the presence of *light-blue* indicates that the Contestant did *not* get the question correct.
Next, if the sequence of colours is *red, green, red*, it can be inferred that the Chaser got the question correct. If the sequence of colours is *green, red* and it is the **first** two colours in the array, it can be inferred that the Chaser answered correctly. Similarly, if the **last two** colours in the array are *red, green*, it can be inferred that the Chaser was able to answer correctly.
The results of this inference, i.e, whether the Chaser and/or Contestant were able to answer correctly, is returned as a pair of Booleans (True/False, corresponding to Contestant and Chaser, respectively).

Thus, the development of the first component, comprised of three functionalities, was completed. Following this, a testing suite was constructed, which tested all three facets of functionality in conjunction.
This was done through reading in a video, frame-by-frame, and passing every third frame to the *logic* function. This function would then check the RGB values of pixels at a specific y-coordinate, between two preset x-coordinates, by passing in each pixel to the *colour_checker* function. This function would then return whether a pixel is *light-blue, dark-blue, green, red* or *other*. If other, the pixel would be skipped, and the next pixel would be evaluated, until a match was found. This match would be appended to an array, and then the next pixel would be evaluated. If the same colour was matched, consecutively, or it was not a relevant colour, the pixel would be skipped. This would continue until the end of the preset x-coordinate range.
The resulting array of all identified colours would then be passed to the *inference* function, which would examine the colour patterns to identify whether the Chaser and/or Contestant were correct. This would then be returned as two Booleans.

It is important to note that **if** the resulting array was **less than** three colours or **more than** five colours long, the *inference* function would identify that an error had occurred and exit. This is because the most complex information array - Contestant got answer wrong (presence of light-blue), Chaser picked the middle choice (two red bars), dark-blue Answer box and green correct Answer box, is the *maximum* number of colours that can be inferred. Similarly, if there are less than three colours - green correct Answer box, Chaser picks right/left Answer box (one red bar) and Contestant gets answer correct (no light-blue), it must mean that not

enough colours have been accurately identified, which indicates there is something wrong with the component.
Hence, this error handling made it easier to ensure there were no false positives, i.e, frames with similar colour patterns that were misidentified, as even if there were similar colours, it would be difficult for the pattern and number of colours to fall within the range of acceptance.
Another important observation from the testing suite was that when there were two consecutive dark-blue boxes, the program would not update the colour, as there was no change within the colour. Hence, while the *colour_checker* function does not explicitly identify this as *two* separate boxes, the *inference* function is robust enough to identify that the Contestant must still be correct (no presence of light-blue), and identify whether the Chaser is correct, dependent on the placement of the green and red colours.

8.2.2 Question Frame Component

There were some key issues that were identified after the development of the Answer Frame component.

Firstly, the program would identify *all* frames that matched the criteria defined above, i.e, if a frame had the red/light-blue/dark-blue/green colour palette in the specified area. As this information could be present on screen for multiple seconds, the program would extract and analyse *all* of these frames and register all of these responses as distinct questions.

Additionally, the program also had no way of distinguishing between Questions, i.e, if the Chaser and Contestant both answered Question 1 and Question 2 correctly, the program would have no way of flagging that these were two *unique* questions, as the array of colours extracted by the program would be the same.

Thus, it was imperative to identify a distinguishing event that allowed the program to identify when the video had moved from one question to another. This was the first step toward tackling the aforementioned issues.
One such unique event is the **Question Frame**. An example of this frame is presented below:

Figure 13: Question Frame

As seen above, this frame poses a Question but **does not** have the Answer Boxes on screen. The most important property of this frame is that it **only appears once** per Question, i.e, once it appears, it stays on screen until the Answer Boxes appear or the Offer screen is presented. Thus, this is a distinguishing event that allows the program to flag when a new Question has begun.

Thus, in order for the program to identify this frame, a similar approach to the Answer Frame identification was employed. As the Question Box appeared in the same place throughout each video (as the videos were obtained from the same source), it was simple to identify and set the region that the program was required to check. If the RGB values of the pixels within the checked region were the same as a certain range of preset values (that corresponded to a specific shade of blue), the program could pinpoint the frame as a Question Frame.

Once this Question Frame was identified and extracted by the program, it acted as a distinguishing event between Answer Frames, and made it easier for the program to identify which Answer corresponded to which Question.
However, this still did not solve the problem of many consecutive Answer frames being detected.

Hence, in order to solve this, a *buffer method* was utilised. This method utilised an array of

length 1, which would hold an Answer Frame. Simply put, whenever a frame corresponded to the criteria of the Answer Frame component (specific colour palette and appropriate array size), it would be placed into the buffer. Thus, each identification of an Answer Frame would overwrite the one placed into the buffer **until** a Question Frame was identified.
Once a Question Frame was identified, the buffer would then be extracted and analysed, and the appropriate result (whether the Chaser and Contestant got the question correct), would be obtained and written to an array. Once this was done, the program would erase the buffer and restart the whole process.
Thus, this method enabled the program to *only* analyse and write the results of the **last** Answer Frame corresponding to each **distinct** Question.

8.2.3 Offer Frame Component

The final Component of this section is the **Offer Frame Component**. As mentioned earlier, one of the key issues identified with the Answer Frame component was its inability to identify *which* Question corresponded to which Answer Frame.
Similar to this issue, the Question Frame component proposed a related concern, that the current implementation made it impossible to identify *which* Contestant each set of Question Frames pertained to. As each episode consisted of four Contestants, it was extremely important to identify which Contestant was answering which set of Questions, in order to identify the win/loss rate, the amount contributed to the prize fund, etc.

Thus, similar to the previous section, it was important to identify a distinguishing feature that differentiated Contestants from each other.
The most obvious choice of a unique event was the Offer Screen, that depicted the amount of money presented to each Contestant by the Chaser. An example of an Offer Screen is presented below.

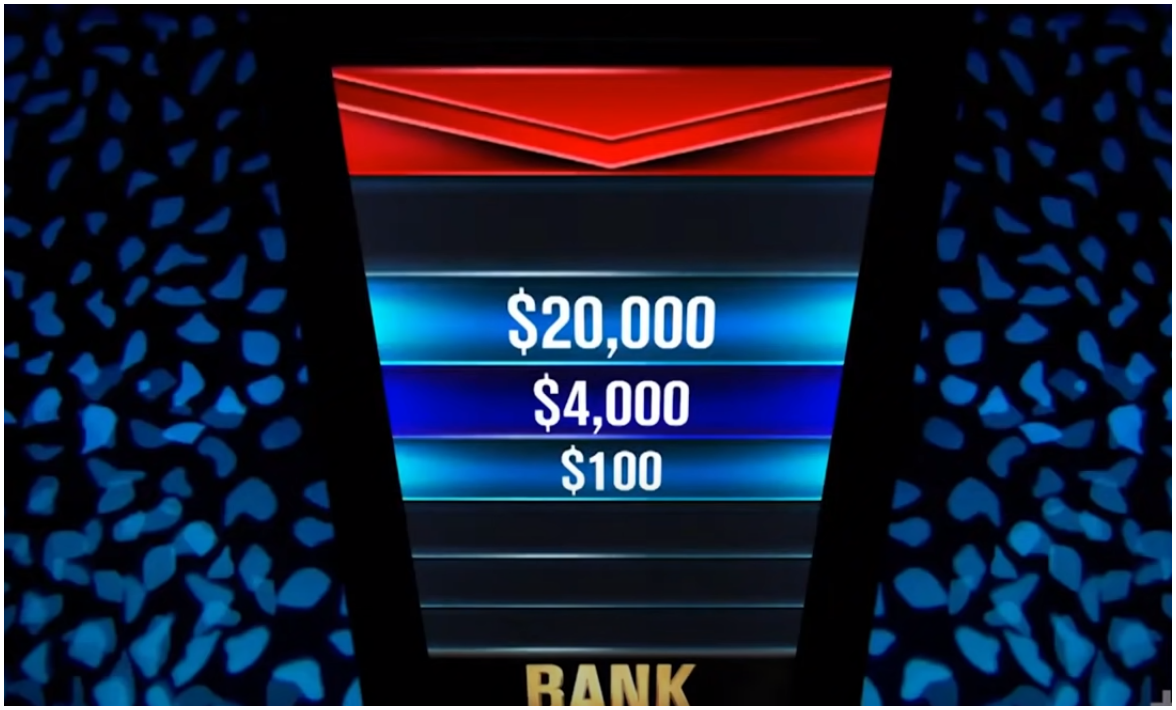

Figure 14: Offer Frame

This screen, like the Question Frame, is also only presented once per Contestant, i.e, once this Offer Screen appears, it does not appear halfway through the Question Set again, as this frame is almost immediately followed by the Contestants' selection of an Offer, which is then interspersed throughout the Question Set.
Unfortunately, unlike the Question Frame, this Offer Screen does *not* stay on screen consistently, and is often prone to cutaways. Thus, the solution adopted for the Question Frame would *not* work for this, as the **same** Offer Screen could appear multiple times, with breaks in-between, making it difficult to use this as an isolated distinguishing event.
Thus, the previous solution needed to be modified slightly, through setting a mandatory number of frames that must have elapsed between the first Offer Screen grab, and the next. This was calculated using an approximation of the length of time a Contestant would be answering questions - roughly five minutes, or three hundred seconds. As the video was 30FPS, and the program was only analysing every third frame, this meant that roughly three thousand frames needed to elapse, before the program could evaluate the Offer Screen for the next Contestant.
Thus, this method, in conjunction with the previous solution, enabled the program to obtain the Offer Screen presented to each Contestant, which demarcated each Contestant's turn.

The actual identification of this frame was conducted through similar techniques utilised for the Answer Frame and Question Frame components, i.e, checking the RGB values of pixels in a specific area. However, as evident from the example, it is clear that there are a variety of shading elements incorporated within the top and bottom strips containing a monetary value. Thus, when setting the preset range of RGB values that each pixel could take, in order to qualify as an Offer frame, there were some issues faced.

Setting the range of preset RGB values that qualified for an Offer frame too broad would result in false positives, causing issues for the OCR (discussed below). The distinguishing event functionality of the Offer frame was also compromised. Thus, in order to address this, instead of using a *strip*, the program used an *area* of pixels that were examined, and the average value of this area was computed, in order to evaluate whether the composition of pixels qualified to be an Offer frame. This method facilitated the contraction of the range of preset values for the top and bottom strips, which made it more difficult to obtain false positives.
Furthermore, a frame would only pass as an Offer frame if, alongside *both* top and bottom strips' areas qualifying as a specific average blue colour, the *middle* strip would also correspond to a specific colour. As the middle strip has minimal shading, this was much easier to compute, as the range of preset values were much smaller.
Once these functionalities were tested independently, there were utilised in conjunction, to obtain the appropriate Offer frames.

8.2.4 Text Recognition Component

Alongside being a distinguishing event, it is also important to note that the retrieval of this Offer frame had another critical use, i.e, extracting the monetary values of the Offers presented to the Contestant by the Chaser, using an Optical Character Recognition tool, i,e, *PyTesseract*.
Once the Offer frame was obtained, this frame was then passed into a function, *numbers()*, which utilised the in-built Open-CV2 tools to apply filters and thresholding transformations to the supplied image. The filtered image was then "read" by PyTesseract, in order to return the monetary values presented on screen.
The combination of filters and transformations applied, was mainly trial and error, as it was difficult to predict which blend of configurations which worked. Ultimately, the final combination ended up being a synthesized image obtained from the following:

- Crop: the input frame was cropped to the appropriate dimensions, in order to minimize noise.
- Mask: Removed pixels outside of a specific range of RGB values, to further reduce noise.
- Morphological Transformation: Pixels near the boundary of the image are discarded, further removing noise.
- Dilation: As the image has just been eroded to remove white noise, the subject of the image, i.e, the numbers have shrunk. Thus, dilation increases the size of the image, as they are the foreground object, and also rejoins any broken parts.
- Thresholding: Convert image to black and white.

An example of this configured image is presented below.

Figure 15: Filtered Image

The resulting text, as 'read' by the program is also presented below.

```
$20000

$4000

$100

['$100' '$20000' '$4000']
```

Figure 16: Image Output

Once this image had been obtained, the program was able to obtain the monetary values much more easily through using the PyTesseract *image_to_string* function, with configuration 12. This configuration basically aims to identify as much text as possible, in no particular order.

Thus, through the combination of this configuration and the combination of filters, the process of recognising the monetary values presented on screen, became much more streamlined.

8.2.5 Chosen Offer Component

The final component that was created was the *Final Offer* component. Once the Offer frame had been obtained, the next relevant frame was the frame that indicated *which* Offer was chosen. As either the Top, Middle or Low Offers could be picked, the top, middle or bottom strips could all go green. Hence, it was a simple check to identify whether a given set of pixels in a specific location were all green.
Like the Offer frames, an area functionality was utilised, as this was only one colour, and it made more sense to evaluate an area of pixels and obtain the average colour. If the average colour was within a specific set of values, it flagged the frame as as Chosen Offer frame, and depending on the specific area of the pixels, the program could extrapolate which Offer was chosen.

8.2.6 Connecting Components

Once all the components were developed, and thoroughly tested, the final step was to connect all of the functionality.

This was done through the creation of a state machine, which allowed the program to move between three states - the Offer state, the Option state and the Question state.
The starting point of the program, whenever a new video was supplied, was to start at State 0. From this point, the program would begin parsing each third frame, looking for the Offer screen. Once this was found, the program would extract the monetary values of the Offers presented. The Offer frame was then passed into the *numbers()* function, which returned the monetary values presented on the screen. This was then written to file.

Following this, the program moved to State 1 - the Option state, where the next deployed component was the Chosen Offer component, which evaluated which Offer had been chosen by the Contestant. This was also written to file.
Once the program was able to identify the Offer, it was intuitive to extrapolate the number of steps required for the Contestant to secure the prize fund, i.e, the number of steps to *Home*, as denoted in Figure X.XX. For example, if the Contestant picks the middle/top/low Offer, they are 5/6/4 steps away from Home, and the Chaser is always 8 steps away. Hence, an array of two elements is created, with the 0^{th} element referring to the Contestant's position and the 1^{st} element referring to the Chaser's position.

Following this, the program moved to State 1 - the Question state, which evaluated whether a given frame was a Question frame. If yes, the program utilised the buffer method, to store all the subsequent Answer frames, until the program discovered another Question frame. Once this frame was identified, the frame in the buffer, i.e, the last frame that contained information about whether the Contestant/Chaser was right or wrong, was analysed.
Depending on whether the Contestant is right or wrong, the array from State 1 changes, i.e, if the Contestant is correct, the value stored at the 0^{th} index position goes down by 1, and if the Chaser is correct, the value stored at the 1^{st} position also decrements by 1. If either the Chaser or the Contestant get the answer wrong, the value in the relative array position stays the same.
The end of the state is determined by whether the Contestant's element reaches 0 first, i.e, they get the appropriate number of questions correct, or whether the value of the Chaser's element becomes equal to the Contestant's, i.e, the Chaser catches the Contestant. Once either of these events occur, the result (Chaser Wins or Contestant Wins), is written to file, and the program moves back to State 0, to evaluate the next Contestant, or analyse a new video.

8.3 Data Analysis

Once the relevant information had been extracted from the videos, and written to file, the next step was to set up the analytical pipeline.

The tools used for this were *matplotlib, pandas, statsmodels.formula* and *sklearn.*

These tools were critical in exploring the relationship between the constituent variables and evaluating their importance. Their full feature set and use cases will be explored in the following analysis section.

9 Data Analysis

This section will describe the process and results of the data analysis process described in earlier Sections.

9.1 Data

The main source of data for this project was in the form of videos of *The Chase Australia*, obtained from Vimeo and YouTube. Seventy-two (72) videos were utilised for the purposes of this project, which resulted in a total of two hundred and eighty eight (288) contestants' decisions being extracted by the program. As described earlier, the data utilised for the statistical inference has been obtained from the *image processing* conducted in the earlier parts of this project, and written into a CSV file.
The CSV file consisted of the following information - *episode, high choice, middle choice, low choice, offer chosen by Contestant*. This file was subsequently read in as a *data-frame*. Each comma separated heading corresponded to the creation of a new column, and each column was populated by the relevant information from the text-file.

The following details have been obtained from the CSV file:

- Episode Counter: The order in which the videos have been processed. This is done to maintain integrity of the details extracted, as it enables the program to keep track of which Contestants are part of which episode. Typically expressed in the form 'XYZ.mp4'.
- High Offer: The high Offer presented to the Contestant, by the Chaser, as extracted by *PyTesseract*. This is a numerical value, stored as an integer.
- Middle Offer: The middle Offer presented to the Contestant, by the Chaser, as extracted by *PyTesseract*. This is a numerical value, stored as an integer.
- Low Offer: The low Offer presented to the Contestant, by the Chaser, as extracted by *PyTesseract*. This is a numerical value, stored as an integer.
- Choice: The Offer accepted by the Contestant. This is a numerical value, computed as 1 (High Choice), 2 (Middle Choice) and 3 (Low Choice).
- Win: Whether the Contestant is able to secure the prize fund and go *Home*. This is expressed as a String and can take on two values - 'Contestant Wins' or 'Chaser Wins'.

Following this, the following values could be *computed* from the provided information:

- Order: The ordering of each Contestant within a video. This value is expressed as an integer, and can take on any value between 1 and 4, to indicate the order of their turn.
- Wins to that Point: The number of Contestants that have been able to secure the prize fund *before* the Contestant goes on. If the current Contestant wins, they are *not* included in this metric. Effectively, this is the number of Winners that the Contestant is aware of, before their turn. This is expressed as a float, and is computed.
- Total Wins: The number of Contestants, *including* the current Contestant, that have been successful in securing the prize fund. The current Contestant *is* included in this metric, if they win. Expressed as a float, and is computed.
- Prize Fund: Total money that has been contributed to the prize fund, i.e, the 'pot'. Expressed as an integer, and computed by checking whether the previous Contestant has won, and if so, the program adds their winnings into the prize fund. The first Contestant always starts at 0.

Once this information was obtained, the next step was to conduct some preliminary analyses and provide some intuitive data visualisations.

9.2 Visualizations and Trends

As mentioned earlier, there are a number of hypotheses and relationships that we are investigating in this section.
In this section, we will illustrate the different relationships that we have found.

9.2.1 Probability of Winning

From our data-set, the *win* variable and *order* variables can be used to infer the likelihood of a Contestant securing the prize fund, depending on their choice.

This is done through simply dividing the number of Contestants who picked the Middle Offer and *won*, over the total number of Contestants who picked the Middle Offer. The *.loc()* functionality of data-frames was extremely useful for this.

Out of the 288 contestants, the program is able to estimate that **64.23%** of Contestants that pick the Middle Offer are likely to secure the prize fund, while Contestants that choose the

High Offer only have a **30%** chance of winning. Contestants who pick the Low Offer have a **58.82%** chance of winning.
It is important to note, that out of the 288 contestants evaluated, the vast majority of contestants opted for the *Middle Offer*, thus, the small sample size of individuals choosing the High/Low Offers can inflate the reported win estimations. However, it is still very interesting to observe that only 7% of contestants opt for anything *but* the Middle offer.

9.2.2 Distribution of Choice

Next, the data set offers some insight into the distribution of accepted offers. This is presented below.

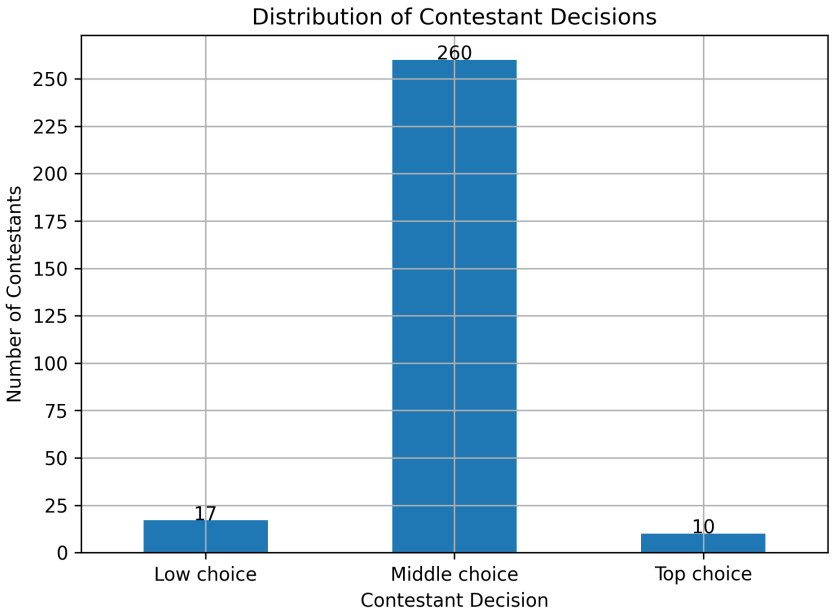

Figure 17: Contestants' Offer Decisions

From the above figure, it is clear that there is a disproportionate number of Contestants who opt for the middle offer. While this may have been obvious, the fact that there is *such* a discrepancy amongst the offers picked, is *quite* shocking.
In the following sections, the specific characteristics of contestants who choose the different offers are explored (ordering, number of questions correct, etc), in order to gain some insight into whether there are some common characteristics between Contestants who opt for the different offers.

9.2.3 Average Reward

From the data, the average monetary value that each contestant is able to secure, can be inferred. This result is presented below.

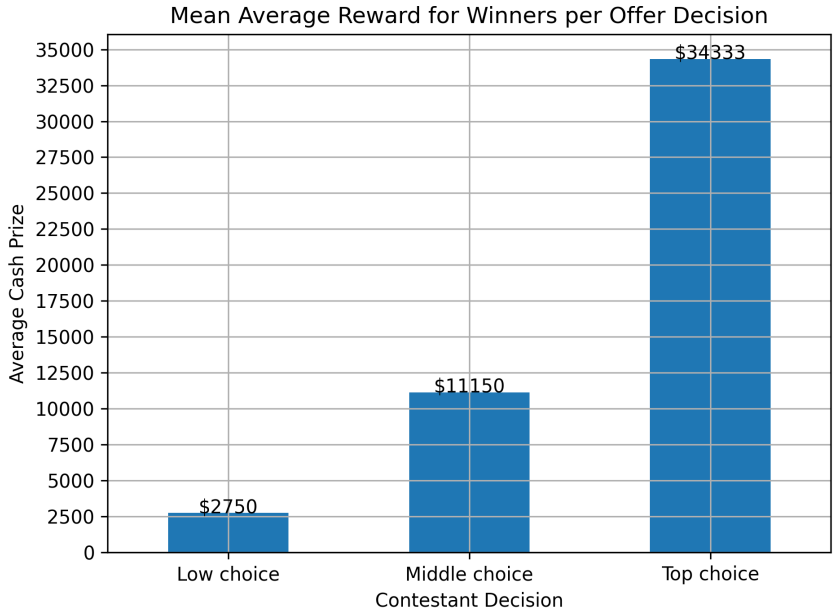

Figure 18: Average Reward

Naturally, the average winnings from the *high offer* are much larger than the average winnings when the low or middle offers are picked. However, the sample size for the number of Contestants who have opted for the high offer is relatively *much* lower, in comparison to the Contestants who opted for the low/middle offer.
However, this is still an interesting metric.

9.2.4 Ordering of Contestants

The next relationship that was observed was that of the *ordering* of contestants, and their win-rate. Similar to the previous, this section deduces the likelihood of a Contestant securing the prize fund, depending on their ordering.

This is done by dividing the total number of winners per ordering each by the total number of players per ordering.

The results for this were quite interesting. The program was able to estimate that **63.89%** of Contestants who went first, were able to secure the prize fund, followed by **51.39**% of winners amongst the Contestants who went second. Next, the Contestants who played third faced a win-rate of **69.44**%, and finally, the Contestants who went fourth were met with a win-rate of **66.20**%.

From this analysis,it is evident that a Contestant who faces the Chaser *second*, has the least chance of securing the prize fund, whereas the Contestant who faces the Chaser *third*, has the highest chances of securing the prize fund.
This is very interesting, as it facilitates exploration of a series of hypotheses about *why* this could be the case, and also signifies that there is *some* relationship between the ordering of Contestants, and their likelihood of winning. It is interesting to question how the decisions made by the contestant may have played a factor in the outcomes that are currently being observed.

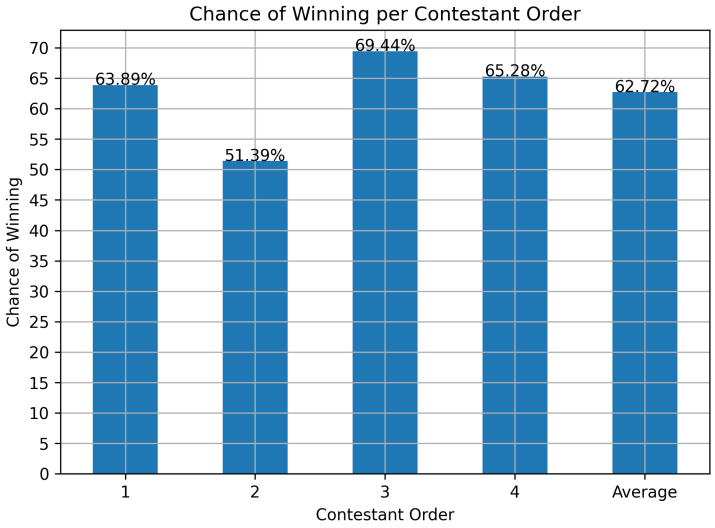

Figure 19: Likelihood of Winning by Contestant Order

9.2.5 Ordering and Choice

This section investigates the relationship between the *ordering* of a Contestant, their choice of Offer, and whether they win.
This is done by dividing the number of winners of a certain ordering, over the total number

of players with the same ordering and the same chosen offer. This yielded some interesting results, presented and discussed below.

If the first Contestant chose the Middle Offer, they had a **63.77**% chance of winning. If they chose the Low Offer, they had a **100%** chance of winning, whereas if they chose the High Offer, they had a **0%** chance of winning (1 Contestant within this category).
Next, if the second Contestant chose the Middle Offer, they had a **53.97%** chance of winning, whereas they had a **67%** chance of winning, if they picked the Low Offer. However, they had a **16.67%** chance of winning, if they picked the High Offer. There were only six Contestants who met this criteria, however, so this estimate could be inflated.
Following this, the percentage of winners who were third Contestants and chose the Middle Offer, was **68.18%**, followed by a 83% of Contestants picking the Low Offer. There were no Contestants in this category who picked the High Offer.
Finally, the percentage of winners who picked the Middle Offer and also went last, faced a **70.97%** chance of winning, followed by a **66.67%** chance of winning when choosing the Low Offer, and a **16.67%** chance of winning when picking the High Offer.

9.2.6 Range of Offers

The data-set also contains some useful information about the range of offers that are presented to the Contestant, by the Chaser.
As mentioned earlier, the middle offer is obtained from multiplying the number of correct answers in the Cash-Raise round, by $2,000 (the reward for each correct answer). Hence, this middle offer is not controlled by the Chaser. However, the low and high offers are presented on the basis of this middle offer, as the low offer -as the name suggests- must be lower than the middle offer, the high offer must be higher.

Hence, the data-set was utilised to find the *deviation* of the low and high offer, from the middle offer. This was done simply, by plotting the middle offers against the difference between the high/low and middle offers. This allows for the inference how much lower the Chaser's low offer can be, given a fixed middle point, or how much higher the Chaser's high offer can be. These visualisations are presented below, followed by a brief explanation.

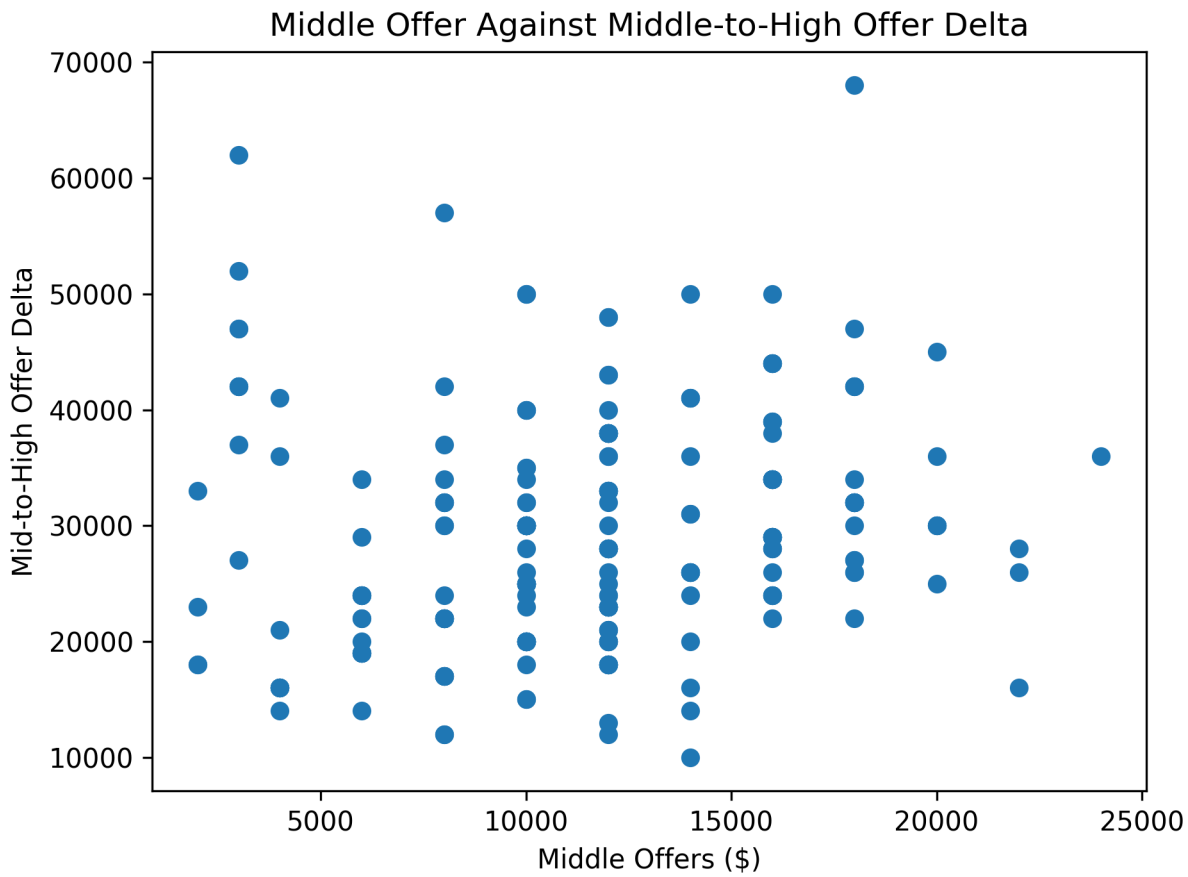

Figure 20: Middle Offer Against $\delta High/MiddleOffer$

This graph illustrates the deviation between the independent middle offer of a Contestant, and the consequent high offers presented.

As seen, there is a fair amount of deviation in this graph, especially in the $10,000 to $18,000 range. This indicates that there is not *much* standardisation to be expected, when a Contestant answers 5 to 9 questions correctly, as the Chaser can offer anything from $10,000 to $50,000 above the middle value.

Something that *is* interesting, however, is the tendency of the Chaser to offer (approximately), $22,500 more than the middle offer as a starting value, when the Contestant gets a relatively higher number of answers correct (8+). This is interesting, as one would expect that a strong performance from the Cash-Raise round might make the Chaser more nervous and cause them to not offer them a high amount of money, lest they accept this offer and win. But this does not seem to be the case, as the Chaser seems to want to goad the Contestant into accepting this tempting offer, and this reduces their head-start, which allows the Chaser to have a better chance of catching up.

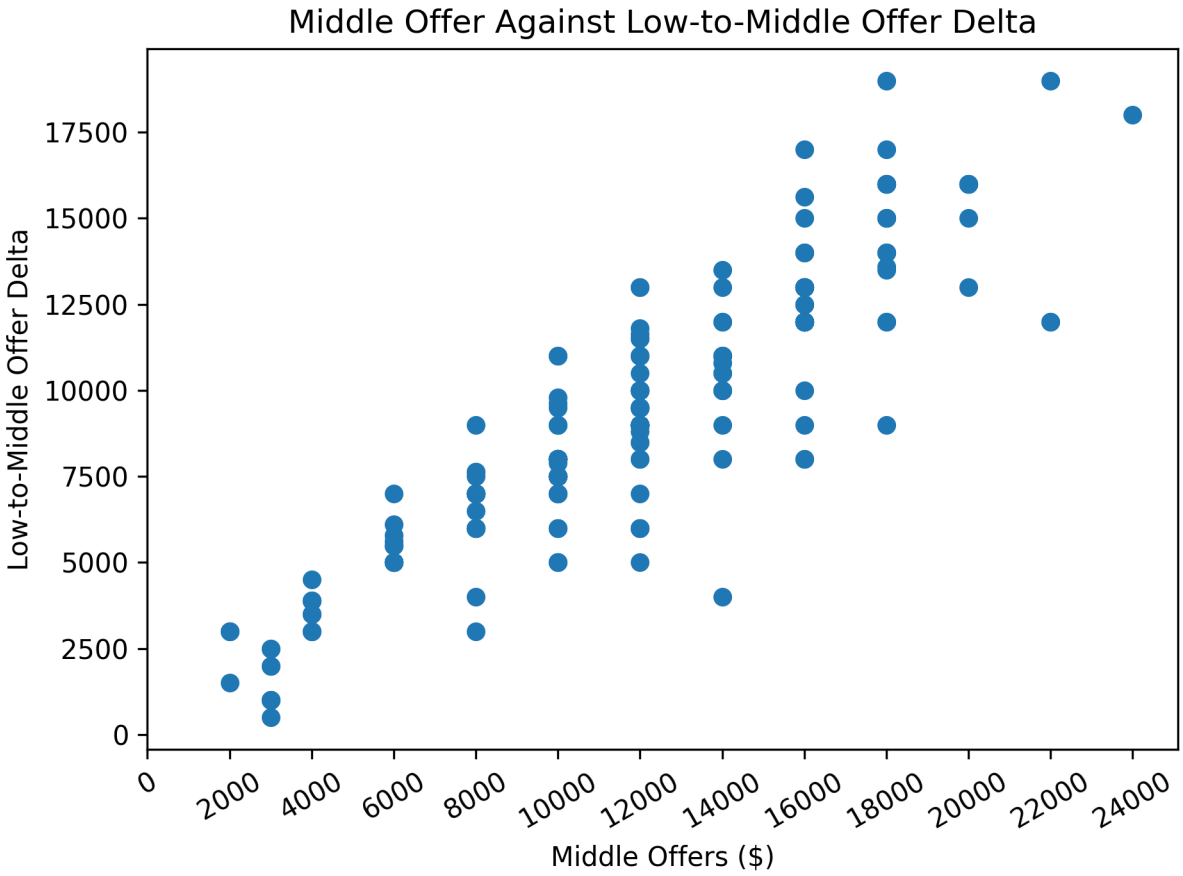

Figure 21: Middle Offer Against $\delta Low/MiddleOffer$

This graph illustrates the opposite of the previous graph- it shows the deviation between the independent middle offer of a Contestant, and the consequent low offers presented.
At first glance, this graph is much more concentrated and predictable than its predecessor. The lowest offer that *tends* to be presented is -$1,000. However, this indicates that this graph does not *really* have the explanatory power that we are looking for. Unlike the previous graph, the low offer graph *does* have more of a limitation in the values it can take, unlike the high offer graph, as the range of low offers is limited n-1 to -$1000, with n being the middle offer. However, while this limitation is present in the data-set, it is not necessarily a characteristic of all the low offers presented on *The Chase*, as the lowest offer *ever* presented on the show has been -$17,000. While this shows that the minimum low offer cannot always be within the range that we presented earlier, it can still be safely inferred that a low offer of -$17,000 is not *really* the norm, and as such, can be discarded as an anomaly.

9.3 Methods & Findings

9.4 Linear Regression

The main objective of this project was to find the statistical significance of constituent factors affecting whether a Contestant chooses a High, Middle or Low Offer, as presented by the Chaser.
In order to do this, a *linear regression* was designed. This was done to model the relationship between the dependent variable (the Offer choice) and the explanatory variables.
The dependent variable for this regression was the *monetary value* of the offer picked by the Contestant. This was computed based on the *choice* of the Contestant, i.e, if the Middle Offer was chosen by the Contestant, the corresponding *middle offer monetary value* was used. This variable is denoted as *choice* for the remainder of this discussion.

The explanatory variables used for this regression were as follows:

- *order*- order of the Contestant, i.e, whether they are 1^{st}, 2^{nd}, 3^{rd} or 4^{th}.
- *wins_to_that_point*: The number of Contestants *before* the current Contestant that have been able to secure the prize fund. This does **not** include whether the current Contestant wins or not. This is expressed as a number between 1 and 4.
- *mid*- the presented Middle Offer. This is an integer.
- *high*- the presented High Offer. This is an integer.
- *low*- the presented Low Offer. This is an integer.
- *fund*- the total amount in the *prize-fund* when the Contestant's turn begins. This is an integer.

Thus, the tested linear regression equation is:

$$\hat{choice}_i = \hat{\beta}_0 + \hat{\beta}_1 high + \hat{\beta}_2 mid + \hat{\beta}_3 low + \hat{\beta}_4 questions\_correct + \hat{\beta}_5 order + \hat{\beta}_6 wins\_to\_that\_point + \hat{\epsilon}_i \quad (2)$$

Once the dependent variable and its subsequent explanatory variables were defined, *statsmodels*' in-built linear regression function was called. As mentioned earlier, this project used OLS as it is BLUE (best, linear, unbiased estimator).
The results of this regression are presented in the Appendix as A.1. However, the coefficients of each variable (β)is reported and discussed below.

β	Coefficient	P-Value	Significance
High Offer	0.1927	0.113	NO
Middle Offer	0.9645	0.000	YES
Low Offer	-0.02595	0.159	NO
Order	-37.1099	0.929	NO
Wins To That Point	-12.1011	0.987	NO
Fund	-0.0095	0.819	NO

From these results, we can see that there are *no* statistically significant variables apart from the middle offer. This is interesting, as the middle offer is based on the number of questions that the Contestant got correct in the Cash-Raise round, and is the only factor that the Contestant has any control over.
The coefficient on the other variables also posit some interesting theories. For instance, the positive and negative coefficients on the high and low offer, respectively, are to be expected - the higher the High Offer, the more tempted Contestants are to take it, and vice versa for the Low Offer.
Interestingly, the *order*, *wins to that point* and *fund* variables all have a negative effect on the Contestant opting for higher offers, i.e, the more money there is in the pot/the more Contestants who have been able to make it *Home* and the later the Contestant's turn is, the less likely they are, to make risky decisions. Intuitively, this makes a lot of sense, as once the money in the pot has begun to build up, Contestants' aversion to loss increases, as there is a lot more to lose, and they are more likely to opt for safer decisions.

Finally, it is important to note that the R^2 value for this regression is 0.37, which indicates that this model can explain 37% of the variance experienced by the dependent variable (choice). This indicates that the model does not perfectly fit the patterns exhibited in the data.

9.4.1 Use of Dummy Variables

The use of dummy variables (variables that take on the value of 0 or 1, depending on whether a Contestant is 1^{st}, 2^{nd}, 3^{rd} or 4^{th}, etc. was considered.
If this approach was taken, only **three** dummy variables would have to be utilised, as a Contestant's position could be inferred depending on the values of the other variables. For example, if a Contestant was 4^{th}, and there were three dummy variables indicating whether a Contestant was 1^{st}, 2^{nd} or 3^{rd}, **all** these variables being **0** would indicate that the Contestant is 4^{th}, without having an explicit variable to indicate this. This is done in order to prevent

issues of *multi-collinearity* or *dummy variable trap*. Thus, by dropping one of the categorical variables, we allow the dropped category to act as a *reference* variable, with the others representing the deviation from this.
An example of how these values would change, depending on order, is presented below. A modified linear regression to accommodate this is also presented.

Dummy Variable Values				
Order	Dummy 1	Dummy 2	Dummy 3	(Inferred) Dummy 4)
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

The modified linear regression would be:

$$choice_i = \beta_0 + \beta_1 high + \beta_2 mid + \beta_3 low + \beta_4 questions\_correct + \beta_5 wins\_to\_that\_point + \beta_6 order1 + \beta_7 order2 + \beta_8 order3 + \epsilon_i$$

However, we chose to not represent the *order* variable in this manner, as this would indicate there is **no** relationship between the order of Contestants and there is no reason to preserve the ordering.
However, there *is* an order- the first Contestant *does* have more of a relationship to the second Contestant, than they do with the fourth Contestant. Therefore, this approach was not adopted for the purposes of this project.

9.5 Logistic Regression

In this project, a *logistic regression* was also utilised, in order to model the probability of a particular Offer being chosen. This was done through splitting the data into *testing* and *training* data. Training data was 20% of the overall data-set.
The variables utilised for this logistic regression were the same as those used in the *linear* regression: *choice, high, mid, low, wins_to_that_point, order*, with *choice* being the dependent variable.
A scaling function was also added to this data, unlike the linear regression, in order to scale the input variables into a standard range, as a large spread of values may cause large error

gradient values.
The *sklearn* LogisticRegression.fit() function was used, with an L2 penalty term, and with a maximum of a 1000 iterations.
The logistic regression conducted in this project was *unbalanced*, as the majority of Offers chosen by the Contestants were the *middle* offer. As such, when running a logistic regression for this imbalanced data set, the predictive modelling outcome states that an individual will choose the middle offer with 91% accuracy. While this is extremely high accuracy, it is unsurprising.

However, when the logistic regression is conducted using a *balanced* condition, in order to handle the imbalanced data-set, the accuracy drops sharply to 43%. This is not a particularly relevant finding. As our data-set is comprised of Contestants from various seasons, across different time periods, and with different Chasers, it is reasonable to infer that the majority of Contestants opt for the middle Offer, and this is a characteristic of *The Chase*, and not a fault of the data-set. Due to this feature, an unbalanced logistic regression was a purposeful choice. Hence, if balanced, it defeats the point of seeing *which* coefficients influence this decision, as we are punishing the model under the impression that the data-set is not robust, whereas it actually depicts how people behave.

With that being said, the coefficients obtained from the logistic regression were also explored and analysed. These are presented, in order of importance, below.

β	Coefficient
Middle Offer	1.617
Low Offer	-0.776
High Offer	0.563
Wins To That Point	-0.161
Order	0.157
Fund	-0.013

From the obvious results, some obvious comparisons can be drawn between the coefficients presented as a result of the Logistic Regression versus those drawn from the Linear Regression. Primarily, we note that the Middle Offer still has the most effect on the Offer picked by Contestants, as it is of a significantly larger magnitude than other parameters. It retains its positive effect, as seen in the earlier section.
The magnitude of the high and low offers appear to be switched, i.e, the low offer has a bigger

decisive effect than the high offer, which is the opposite to the phenomenon observed in the previous regression technique. Nevertheless, the effects still remain fairly minor, and the signs remain consistent.
Following this, it can be seen that the *wins to that point* variable also has a minor *negative* effect on the decision-making process, i.e, the higher the number of Contestants who have won previously, the less likely the current Contestant to take on the Higher Offer, and this is to be expected. This is echoed in the *fund* variable, which has the least effect on the decision-making.
However, the biggest difference in this section is that the *order* variable is positive, inconsistent with the previous section. This indicates that the later the turn of the Contestant, the more likely they are to go for a High(er) Offer. This is quite interesting, and as this is the first variable to indicate some discrepancy, it will be investigated in greater detail later.

Finally, it is important to note that the R^2 value for this regression is 0.33, which indicates that this model can explain 33% of the variance experienced by the dependent variable (choice). While this is not extremely high, it is evident that this technique has some statistical power.

9.6 Random Forest Regression

The next technique utilised was the *Random Forest Regression*. As this is a machine-learning model, some research needed to be conducted prior to its usage, as this was a very new concept.
This regression was utilised in a very similar manner to the *Logistic Regression* conducted earlier, i.e, as 20% of the overall data-set was the training set, and that the data was scaled, in order to control the spread of values.
The number of *trees* in the forest were set to 100, with a maximum depth of 25 nodes. For this project, out-of-bag samples were utilised to estimate the generalisation score, i.e, the number of correctly predicted rows from the ensemble.

The results obtained from this technique were quite interesting. The initial set of results presented the predicted values per tree, of what monetary value each Contestant would pick. This result is presented in the Appendix as A.2.
However, the mean predicted value presented by this model was $11,823. Immediately, it is clear that this significantly lower than any High Offer that could be presented to the

Contestant. Thus, it is obvious that this value is reflective of a *higher* Low Offer or a *lower* Middle Offer, which is quite interesting, and exhibits that Contestants *may* tend to be more risk-averse than risk-loving.

This regression, like its predecessor, also yielded some interesting coefficients to consider. These are presented below.

β	Coefficient
Middle Offer	0.659
High Offer	0.159
Low Offer	0.080
Fund	0.040
Order	0.037
Wins To That Point	0.025

It is obvious that there are some immediate comparisons to be drawn. Firstly, the ordering of the importance of the three offers is reminiscent of that presented in the Linear Regression section. As always, the effect of the Middle Offer is always significantly higher than any other variable, implying that this has the most influence over the final amount chosen. There also seems to be a noticeable gap between the effect of the High and Low offer, but as seen from previous regressions, the contributing effect of these values appear to be interchangeable, and as such, requires more examination.

Surprisingly, the remaining three variables, *fund, order, wins to that point* are all positive. This is the first instance of this occurrence, as until this point, *fund* and *wins to that point* have always been negative, illustrating Contestants' loss aversion as the prize fund increases.

However, all three of these variables being the same *makes sense*, as it can be argued that all three of these variables have a compound effect on the Contestant, i.e, as the ordering of the Contestant increases (they are the last or penultimate player), the chances of there being money in the pot (and thus, Contestants who have "won" or made it *Home*) increases. Hence, it makes intuitive sense that these coefficients would move in the same direction.
However, the most surprising part is that all three of these parameters are *positive*, going against the risk-aversion theory that had been implied earlier. Thus, the effect of these variables cannot be confidently estimated, as the Random Forest Regression implies that

Contestants become *more* risk-loving as the money in the prize fund increases. The R^2 of this model was 0.36.

9.7 SVM Regression

The final unique regression utilised in this project was the use of Support Vector Machines (SVM), another machine-learning model that aims to find the hyperplane that is able to distinguish input variables most clearly.
This regression was set up in a very similar way to the Random Forest and Logistic regressions, i.e, 20% of the data was utilised as training data and the inputs were scaled. The StandardScaler.fit() function was utilised for the estimation of the following coefficients:

β	Coefficient
Middle Offer	167.916
High Offer	95.433
Low Offer	68.742
Order	-14.994
Fund	4.179
Wins To That Point	3.317

These results are ordered in a strikingly similar manner to the results obtained from the linear regression. Unsurprisingly, the middle offer continues to be the variable with the most influence, followed by the high offer and then the low offer.
The negative sign on the *offer* variable indicates that this model is able to identify the loss-aversion experienced by Contestants whose turn is towards the end of the game, but does not pick up on a similar phenomenon for the *fund* and *wins to that point* variables. The R^2 of this model was 0.118.

9.8 Polynomial Features

The final statistical technique deployed was the use of *polynomial features*, i.e, interaction terms for all of the variables that were explored in the previous regressions.
This was done by using PolynomialFeatures(), with a degree of 2 (i.e, choose 2 features from the feature matrix), and this was applied to the all of the explanatory variables within the set. This was conducted in order to evaluate whether the combination of factors would have any interesting effects on the decision-making of Contestants.
The results from this exercise were presented in the form of a confidence matrix, which is

shown below.

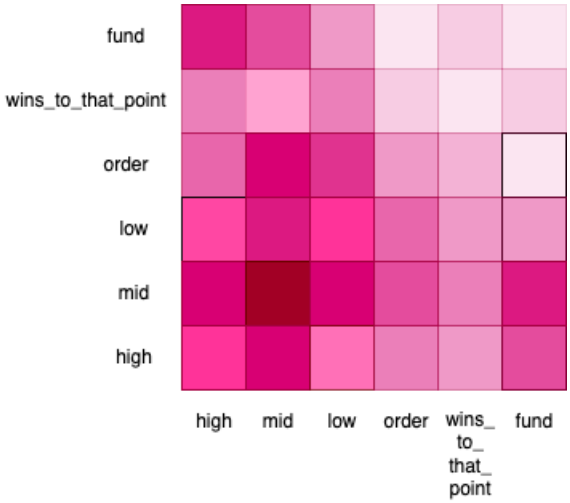

Figure 22: Matrix Frame

This matrix denotes the strength of the relationship between constituent factors. The lighter the shade, the weaker the relationship.

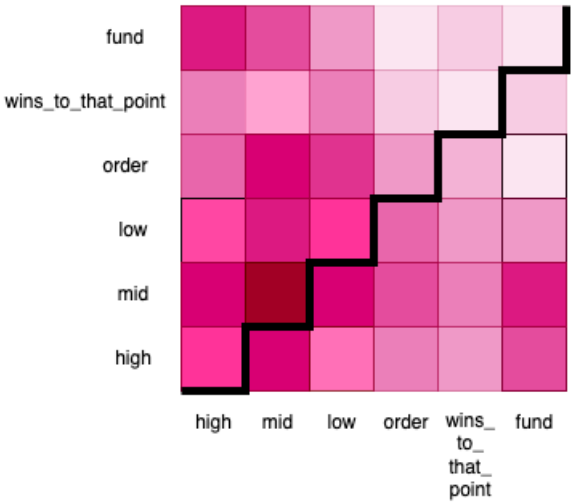

Figure 23: Reflected Matrix

As the matrix is reflected across the y-axis, a line to represent this demarcation has been added to aid visual understanding. Thus, any information below the line can be ignored. From this image, it is immediately visible that the strongest relationships are centered around the *middle* offer, i.e, any interaction with this variable automatically makes the resultant

combination stronger than the other terms.
However, even amongst these relationships, the strongest relationships are experienced by the middle x high term, and the middle x low term.
This is not particularly surprising- as noted in the previous regressions, these three are always the most significant factors in any decision-making process, and as such, it is expected that these represent the strongest relationships. However, more interestingly, the mid x order/high x order/low x order interaction is *more* significant than the mid/high/low x wins_to_that_point term or mid/low/high x fund term, which indicates that the ordering of the Contestants is the next most urgent factor that influences decision making, after the presented Offers.

10 Evaluation and Critical Appraisal

All primary objectives outlined in the DOER were met by this project. The presented artefact is capable of downloading videos from a given URL or playlist link and extracting relevant information using computer vision techniques. Appropriate data analyses tools, such as linear regressions, logistic regressions and data visualisation techniques have also been utilised to conduct statistical inference on the extracted information. This has allowed for the efficient estimation of explanatory variables' coefficients, which has promoted deeper understanding of *which* constituent factors play the most significant role in human decision-making.

The first secondary objective has also been met by this project. This project employs the use of multiple machine-learning models- Logistic Regression, Support Vector Machines (SVM) Regressions and Random Forest Regressions, in order to predict the discrete values of explanatory variables' coefficients. The use of these sophisticated models also serves to deepen the understanding gained from the statistical analyses conducted initially.
The subsequent secondary objective was *not* met by this project, as there as not enough time to implement new image processing techniques, in order to evaluate different game-shows. Thus, this is a point of future growth within the realm of this project.

Overall, this project required the deep understanding and implementation of two key fields - *computer vision* and *statistical analyses*, in order to fulfil the objectives that were set out.
Personally, as I had no experience with *computer vision* and *image processing* techniques, this project served as a meaningful gateway into this realm of programming. I was able to obtain a variety of skills pertinent to this field, such as the deeper analyses of pixel composition and the utility of various configuration techniques to enhance and refine an image. This project was an excellent platform to learn about the skills required to extract necessary information from videos through image analysis and machine learning.

Furthermore, this undertaking was also essential in consolidating my understanding and implementation of statistical techniques and machine-learning models. As a Joint-Honours student with Economics, I had previously only been exposed to digital linear regressions on *Stata*, and as such, it was an extremely rewarding experience to be able to delve into this field further, and explore a variety of other techniques that could be utilised to determine the weights of specific factors on the process of decision-making. Similar to my experiences with computer vision, my experience with statistical modelling within Python and subsequent

machine learning techniques was extremely limited. Thus, this project has been a crucial aspect of my exposure to such a popular and essential field, and for that, I am extremely grateful.

I believe that this project has uncovered some crucial conclusions about human decision-making, despite only evaluating this topic in a constrained scope. The results of the project have been *extremely* interesting, as it has shown that there are very few *significant* factors that influence human-decision making, if any. Primarily, the most (and only!) significant factor appears to be the Contestant's own ability- proxied by the number of questions answered correctly in the Cash-Raise round. This is a very interesting conclusion as it serves as a portal into evaluating the effect of intrinsic factors (i.e, self confidence, self fulfilment, ambition, etc) and the effect of extraneous factors on decisions made by individuals. Thus, this conclusion is quite revealing.

Alternatively, I would have also liked to explore the effects of some more sophisticated parameters on the decisions made by Contestants. Specifically, I would have liked to evaluate the effect of *which* Chaser was being faced, on the decision process, specifically as the seasons went on. I would have liked to incorporate facial recognition technology to do so, and it would be interesting to observe whether the effect of this parameter changed as seasons went on, as Contestants would become more familiar with Chasers' play-styles, strengths and weaknesses.
Additionally, I would have also liked to investigate the effect of the dialogue between the Chaser and the Contestant. It would have been interesting to explore and understand the sentiment behind these conversations, and assess whether there was any effect of these conversations on the choice made by the Contestant, as well as the monetary value of the High and Low Offers presented by the Chaser.
These are some potential directions that future work on this project could be taken into. However, based on the results from the current venture, it is difficult to estimate whether these factors would have any significant effect on human decision-making. Despite this, it would still be interesting to exhaust the analysis of all extrinsic factors on decision making within the Chase, in order to isolate the impact of intrinsic beliefs and understanding.

11 Conclusion

Overall, I believe that the findings of this study are truly quite surprising. From the outset, there appeared to be numerous factors that had the potential to influence contestants' decision making, such as the ordering of contestants, the number of winners, total amount of money in the prize fund, etc. Thus, the unofficial hypothesis was that there would be a multiple significant factors and/or relationships.

However, truth is stranger than fiction. The results of the regressions ran over the course of this project illuminate that there are few significant factors that a contestant would use to inform their decisions of which Offer they pick. It appears that the **only** significant factor that Contestants truly value, is the monetary value of the Middle Offer, i.e, the Offer that the Contestant has "earned" through the Cash-Raise round. By proxy, this postulates that the *only* factor Contestants rely on- is their own ability.

12 User Manual

This program can be run using Python 3.9.

To Download Videos: The program should be run from the terminal as 'python3 main.py download urls.txt'. The 'urls.txt' file is a file comprising of a list of URLs that correspond to videos of The Chase Australia. This will download all the videos to the root folder and will create a text-file of the title of the downloaded videos called 'videos.txt'.

To Extract Information: The program should be run from the terminal as 'python3 main.py analyse videos.txt'. This will extract the information from each video and write it to a text-file 'information.txt'.

To Analyse Data: The program should be run from the terminal as 'python3 DataAnalyser.py information.txt'.

13 Other Appendices

13.1 A.1. Full Regression Results

OLS Regression Results

Dep. Variable:	picked	R-squared:	0.016
Model:	OLS	Adj. R-squared:	-0.002
Method:	Least Squares	F-statistic:	0.8663
Date:	Fri, 18 Mar 2022	Prob (F-statistic):	0.504
Time:	05:42:05	Log-Likelihood:	-23.185
No. Observations:	277	AIC:	58.37
Df Residuals:	271	BIC:	80.11
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.0421	0.071	28.771	0.000	1.902	2.182
high	-2.697e-06	2.07e-06	-1.304	0.193	-6.77e-06	1.37e-06
mid	1.991e-06	4.55e-06	0.438	0.662	-6.96e-06	1.09e-05
low	1.701e-05	1.07e-05	1.595	0.112	-3.99e-06	3.8e-05
questions_correct	9.955e-10	2.27e-09	0.438	0.662	-3.48e-09	5.47e-09
order	0.0157	0.023	0.687	0.493	-0.029	0.061
wins_to_that_point	0.0059	0.029	0.204	0.838	-0.051	0.062

Omnibus:	135.836	Durbin-Watson:	1.839
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1283.973
Skew:	1.740	Prob(JB):	1.54e-279
Kurtosis:	12.957	Cond. No.	7.55e+19

13.2 A.2. Full Random Forest Results

```
[21260.          11980.           8000.           9972.5
 16000.          18000.          16000.           8060.
 12000.          10620.          11980.           3947.
 18000.          12000.          11780.          13350.
 13150.           2612.          11980.          11710.
  3876.           7976.66666667 10040.          11940.
 11360.          11960.           2726.          21440.
 15700.          18000.          13360.           3941.
  2703.          14200.          10280.          11840.
  8040.          11970.           1906.          16000.
 11027.5          3960.          18000.          11355.78571429
 11018.33333333  7992.5           2700.          12130.
 16000.           9700.          17670.           8915.5
 15850.          18000.           2700.           3960.        ]
```

14 Bibliography

[1] "Sony Corp. Of America V. Universal City Studios, Inc. - Wikipedia". Wikipedia, 2022,
https://en.wikipedia.org/wiki/Sony_Corp._of_America_v._Universal_City_Studios,_Inc.
[2] "Pytube — Pytube 12.0.0 Documentation". Pytube.io, 2022,
https://pytube.io/en/latest/.
[3] "Home - Open CV". Open CV, 2022,
https://opencv.org/.
[4] "Pytesseract". Pypi, 2022,
https://pypi.org/project/pytesseract/.
[5] "Pandas". Pandas, 2022,
https://pandas.pydata.org/
[6] "Matplotlib — Visualization With Python". Matplotlib.Org, 2022,
https://matplotlib.org/.
[7] "Github: Where The World Builds Software". Github, 2022,
https://github.com/.
[8] Goldstein, W. (1990). Judgments of relative importance in decision making: Global vs local interpretations of subjective weight. Organizational Behavior And Human Decision Processes, 47(2), 313-336. doi: 10.1016/0749-5978(90)90041-7
[9] Doyle, J., Green, R., Bottomley, P. (1997). Judging Relative Importance: Direct Rating and Point Allocation Are Not Equivalent. Organizational Behavior And Human Decision Processes, 70(1), 65-72. doi: 10.1006/obhd.1997.2694
[10] Summerfield, C., Tsetsos, K. (2015). Do humans make good decisions?. Trends In Cognitive Sciences, 19(1), 27-34. doi: 10.1016/j.tics.2014.11.005