ADS2001 Virtual Internship

Nephrotex

MONASH University

By Isaac Wood, Louise Childs, Aiden Abraham, Yilin Han, Oscar Brazzale

Table of Contents

Executive Summary

Virtual Internships is a company that runs academic simulations for students in the US. The Nephrotex simulation is run based on a bioengineering design challenge. The data that was explored covered 15 implementations of the challenge with 5 groups of varying sizes from 4-7 students. The data that was collected consisted mainly of the chats between groups. This included between members and with mentors. The other part of the data was the outcome score. Students were marked on a scale of 0-8 based on an individual submission they made at the completion of the design challenge, this was the outcome score. The aim of the investigation was to attempt to use the given data to model the outcome score of the students.

The dataset that was worked with was reasonably clean and usable, there were small errors in the data frame that were removed due to the irrelevance to the overall project. From this new clean data two new datasets were formed, one with the contributions of individual users and their outcome scores, and another with the summed contributions of entire groups and the average score per user in that group.

Some initial testing and analysis found that the most common outcome score was 4 with just over 1/3rd of all students receiving that outcome score. It was also found to be incredibly difficult to get the higher scores with only 2.4% of students receiving a mark above a 6. This initial analysis led to the idea that modelling the data may become difficult with bias potentially coming into the model with the same scores being present so often and a lack of higher scores to potentially identify what traits lead to a higher score.

Some of the problems encountered when working with the data were, the inconsistencies in the number of people in each group, and the individual scoring within the same groups. This leads to difficulty associating chats to outcome score which is the initial problem that is explored. To help combat this problem, two different approaches were tested. First approach was to focus on the individuals who received high scores and analyse their chats to see the key ideas/words that were used and see if that had a direct effect on the outcome score or not. The other approach was to focus on the groups who had the highest average score across their members and focus on the content of the chats between the groups. Using this, a key set of words was found and this was used to help initial analysis of data, and later data modelling. This was found using NLP programming, more specifically TF-IDF.

Modelling was the next step in investigating the relationship between the chats and the outcome score. Many models were attempted with importance being placed on the decision tree classifier and the random forest modelling techniques. Due to the output of the model being discrete, these models were deemed most appropriate. Other factors included the binary classifications of the chats that seemed to fit the idea of the classifier models.

A variety of models were implemented in order to try and predict the final outcome scores of the students, and the overall groups, with varying accuracies. From these implemented models the most accurate for predicting individual user score was seen to be a Decision Tree Classifier, returning an accuracy of 0.311. For predicting the average score for a user in a group the most accurate model was seen to be a K Nearest Neighbours model which resulted in an accuracy of 0.370. Other unsuccessful models used included Linear Regression, Random Forest Classification, kMeans Clustering, AdaBoost and SMOTE.

The use of the NLP words was then applied to the models to see if they improved the accuracy. Unfortunately the use of the additional important words in the model did not lead to improved scores across any of the appropriate models.

All of these approaches promised their own limitations due to the nature of the task at hand, the chat data had little to no effect on the group outcome score as outcome score was scored on a group level and a third of the students achieved an outcome score of 4, below average groups were still just as active in the chats as the higher scoring teams and they still discuss the relevant terms. The virtual internship also takes place over 18 hours and many implementations occur at the same time so definitive conclusions are hard to determine. As we were faced with unbalanced data due to the frequency of an outcome score of 4, SMOTE, which is an improved overbalancing technique which utilises data augmentation could be used, this however reduces false negatives and increases true positives at the cost of accuracy, this wasn’t suitable for our analysis due to the low accuracies to begin with, however a future in depth data analysis could investigate measures other than accuracies such as precision and recall when providing analytical advice. The natural language processing analysis also faced limitations as TF-IDF uses frequency of words to determine their importance and doesn’t provide meaning to the words, future data analysis should delve into sentiment analysis so the words in the dataset can be provided meaning and used to draw concrete connections with the outcome score.

Main Body

Brief Summary

Throughout the weeks before the submission of our final report and our preparation for the presentation, every member of the group worked regularly and diligently. Members completed their work on time and contributed effectively to team discussions on Thursdays by offering their own thoughts to the group debate. Despite the fact that some of the group members lacked expertise in specific areas, other teammates stepped up to provide support and assistance. All members had developed valuable skills in both the technical and interpersonal components of excellent team discussion by the end of this project. Overall, everyone in the group contributed to the project's successful conclusion.

Project Description and Details

Background Research

Virtual internships are educational simulations designed to help students acquire the critical thinking, acting, communication, and argumentation abilities needed for a given career field. Our study focuses on Nephrotex, a virtual internship programme that is unique in that interns create teams in an online setting and communicate mostly using a chat application. This arrangement also improves communication between interns and their mentors, whose responsibility it is to answer queries and spark stimulating debates.

In Nephrotex's virtual environment, students work as interns for a biomedical engineering company and are required to collaborate in a team to design a prototype device that can help patients with kidney failure. During the internship, the students were required to conduct preliminary research, understand stakeholder needs, design and develop a prototype, as well as test and evaluate the prototype and rationalise their design decisions.

The design challenge also involves responding to stakeholders in the fictitious company. These stakeholders often have varying demands regarding the 5 key outputs of the project. The 5 key outputs are, Biocompatibility, Reliability, Marketability, Cost and Flux. These 5 outputs are important when analysing the importance of the chats between the groups.

Dataset

In this project, we used a dataset containing chats from 15 implementations of the Nephrotex internship. These chats have been annotated to identify the presence or absence of key concepts related to specific design actions and arguments in engineering discussions. In addition, the dataset also includes a rating of each student's final design report during the internship, which is an important metric for assessing the effectiveness of the internship.

There are 16 factors used as explanatory variables in the investigation:

Features	Explanation
userIDs	A unique id for each student
implementation	A unique id for each implementation
Line_ID	A unique id for each chat utterance
ChatGroup	A string value indicates the team for the first half of the internship, while a numerical value indicates the team for the second half of the internship. Midway through the internship, individual members will change teams
content	The content of each chat utterance
group_id	A non-unique numerical id for the team the chatter was in
RoleName	Whether the chatter was a mentor or a player
roomName	A non-unique name for the internship activity the chatter was participating in.

m_experimental_testing	Talk about using experimental techniques to understand the technical features of design
m_making_design_choices	Talk about choosing a specification or characteristic for a design
m_asking_questions	Asking questions
J_customer_consulatant_requests	Talk about justifying design choices by stating that they should meet or exceed stakeholder requests.
J_performance_parameters_requirements	Talk about justifying design choices by referring to performance parameters or experimental results
j_communication	Talk about justifying design choices by referring to facilitating communication among the engineers
OutcomeScore	A mark from 0 (low) to 8 (high) indicating the quality of the chatter's final design report
wordCount	The word count for the chat utterance

Aim

The aim of this project is to use team-level discussion data from the internship to model and predict final report performance. Specifically, our project will be split into two sections. First, we will transform the data to the team-level statistic, clean it up, format it, and then apply machine learning models and create new features to produce the best possible forecasts. Extrapolate lessons from these prediction models about the relationship between discourse features and report scores in this setting, and look into the mentor's potential influence on these associations. This can provide valuable information to the supervisor, identify potential problems in a timely manner and improve the overall quality of the placement.

Pre-Processing and Data Wrangling

Understanding The Data

To understand the data, we firstly displayed our dataset by calling df.shape() to check the dimensionality of the DataFrame. It’s shown that there are 19180 rows and 17 columns in the dataframe. It has been observed that the dataset size is extremely vast, which makes developing a model with a large number of observations inefficient. As a result, the data must be cleaned before it can be modelled. This will also improve the accuracy and efficiency of a machine learning model. Then we implemented the df.dtypes() to help us to get to know the data types in a dataframe.

```
Unnamed: 0                                   int64
userIDs                                      int64
implementation                              object
Line_ID                                      int64
ChatGroup                                   object
content                                     object
group_id                                     int64
RoleName                                    object
roomName                                    object
m_experimental_testing                       int64
m_making_design_choices                      int64
m_asking_questions                           int64
j_customer_consultants_requests              int64
j_performance_parameters_requirements        int64
j_communication                              int64
OutcomeScore                                 int64
wordCount                                    int64
dtype: object
```

For this image we can see that there are 5 columns containing objects(like strings) and others all contain the integer values. This is useful later on in data cleansing because it informs us about the type of data we're dealing with. This is due to the fact that certain operations can only be performed on specific data types (for example, you can't conduct math on strings or concatenate values). If a column that is supposed to be a number (int64 or float64) has a datatype object, it may indicate that some values are not numbers, possibly due to missing or malformed data. Also many machine learning models require the input data to be in numeric format. Knowing your data type is useful at the pre-processing stage, when you may need to normalise numeric variables. After those, we displayed by calling df.describe() to provide a basic descriptive statistics for our dataset. The output shows that the data was found to be of high quality. One small problem was the overlap of group numbers across different implementations, this was easily solved by creating a unique group ID, using group number and implementation.

During the data processing phase, we encountered an issue with inconsistency in the number of members in each group and the scores of individuals inside the group. This made tying the discussion logs to the final scores more challenging, and it became a problem that we needed to debate and fix at first. To address this, we tested and implemented two distinct ways. In the first approach, we focused on high scoring individuals and analysed their chat logs in depth, aiming to understand whether the key ideas or words they used had a direct impact on the outcome score. In the other strategy, we focused on those groups with the highest average member scores and delved into the content of the chats within the group.Through this approach, we identified a set of key words that played a key supporting role in both our initial data analysis and subsequent data modelling. We identified these keywords by using natural language processing (NLP) techniques, and more specifically, we used the word frequency-inverse document frequency (TF-IDF) method.

Data Cleaning

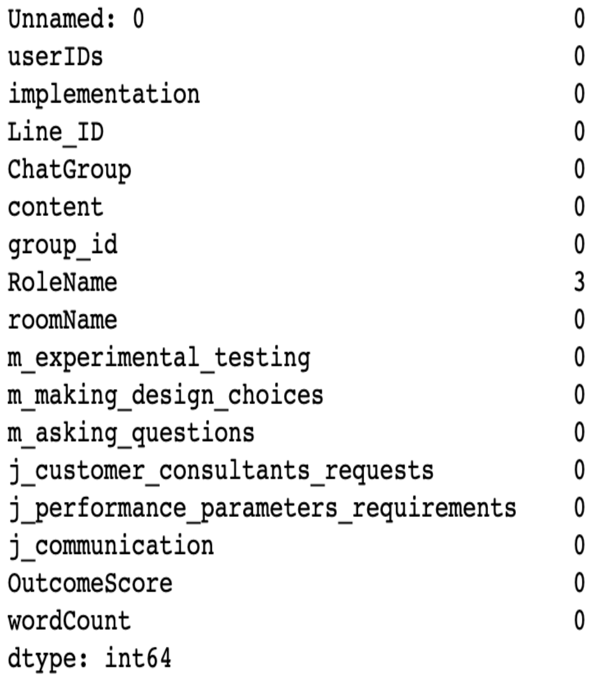
Before deciding which way to approach the missing values within the dataset, we as data scientists must understand why there are missing values. Data scientists have grouped missing data into three categories which includes Missing at Random (MAR), Missing Completely at Random (MCAR) and Missing Not at Random (MNAR). (Swalin, 2018)

1. Missing at Random (MAR): Data may be missing due to external reasons that may not be able to be predicted. The values are just completely missing at random.

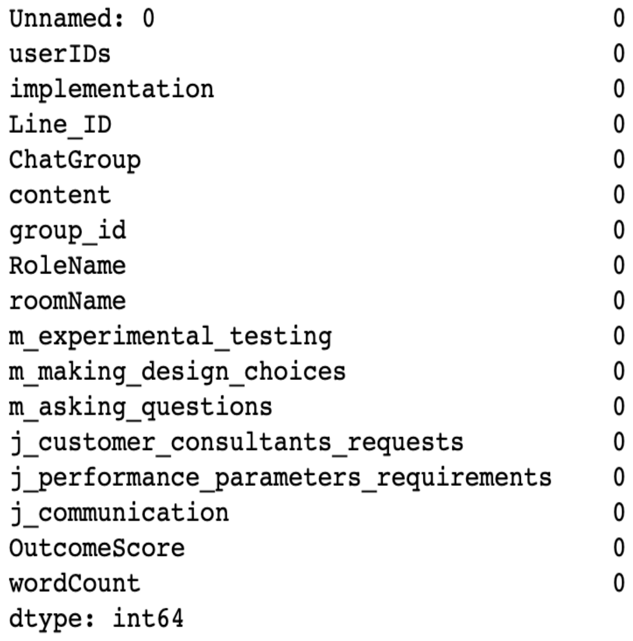
2. Missing Completely at Random (MCAR): Certain values of the data could be missing but have nothing to do with its hypothetical values. For example, people are given weights to weigh themselves for research but some of them are reluctant to weigh themselves. Therefore, the missing data cannot be concluded that it is missing due to observational error.

3. Missing not at Random (MNAR): Not missing at random but are ignorable. The missing value of the data can be determined by the value of interest.

In this case, there may be a number of factors causing missing values in the data.
For example, users (mentors or players) may choose not to participate in certain chats, or their responses may be deleted by mistake or not saved correctly. There is no perfect way to handle missing values in data but what we can do is to minimise it if it does appear.
There are two main ways to handle missing values. One being deletion and second being imputation (Dixon, 2021). According to the number of entries, by calling the function df.isnull().sum(), the output shows that the RoleName got 3 missing values, then we deleted that column. After deleting the columns, we use isnull().sum() on the dataframe to check the number of missing values in each column again, so there are no missing values for the RoleName column.



```
Unnamed: 0                              0
userIDs                                 0
implementation                          0
Line_ID                                 0
ChatGroup                               0
content                                 0
group_id                                0
RoleName                                3
roomName                                0
m_experimental_testing                  0
m_making_design_choices                 0
m_asking_questions                      0
j_customer_consultants_requests         0
j_performance_parameters_requirements   0
j_communication                         0
OutcomeScore                            0
wordCount                               0
dtype: int64
```



```
Unnamed: 0                              0
userIDs                                 0
implementation                          0
Line_ID                                 0
ChatGroup                               0
content                                 0
group_id                                0
RoleName                                0
roomName                                0
m_experimental_testing                  0
m_making_design_choices                 0
m_asking_questions                      0
j_customer_consultants_requests         0
j_performance_parameters_requirements   0
j_communication                         0
OutcomeScore                            0
wordCount                               0
dtype: int64
```

Extracting Necessary Data

In order to eventually start predicting final outcome scores the data needed to be formed into two new data sets, grouped by users and by teams, so that we can attempt to predict the outcome of individual users and the average score of entire groups. This was done by grouping the data frame by either ‘userID’ or ‘Group,’ then finding the sum of this table to find the amount of times each contribution was made. However, for users as the ‘OutcomeScore’ column was now their actual outcome score multiplied by the amount of times they appeared in the chat a new column was needed to be added with this larger incorrect outcome score divided by the number of contributions.
For the group model this outcome score then also needed to be divided by the amount of people within the group in order to get the average score per student which ultimately ranged from 2 to 6.

Exploratory Data Analysis

Method

Given the project's aim and the datasets, some analytical techniques were more pertinent than others. As we are dealing with continuous data and building the relationship between variables, we chose to use techniques such as linear regression, decision tree, and random forest regression. The feature importance will assist to determine which features are directly influencing the model's performance, we will try different combinations of features and assess their impact on the model's performance. Furthermore, we use visualisations including count plots, scatter plots, and heatmap etc. to assist our approach. In addition, we have separated the user model from the group model, which allows us to better analyse individual and collective behaviour and performance separately, which can provide unique insights.

Basic Statistics

The Nephrotex data was based on a virtual internship program that was run. This was an engineering design challenge that was given to groups of interns to work on. There were 15 different implementations of the design challenge with each implementation having 5 groups, and each group containing 4-7 members. At the end of the challenge each individual submitted a design and report that was then marked on a scale of 0-8 with 8 being the highest score. The spread of scores was extremely interesting, it appears to be extremely difficult to get a higher score with only 2.4% of interns scoring a 7 or 8. On the contrary, achieving a 4 was the most common outcome with 33.8% of interns receiving a score of 4.
When the different amounts of each feature were plotted against user score and average group score, as seen in Figures 1 and 2, it was clear that for the individual users communicating about the performance parameters and customer consultant requests had more of a relationship with a higher outcome score. For the group scores it can be seen that making design choices, experimental testing and word count contributed more to an improved final outcome score. However, in all of these graphs it can be seen that there is a lot of variance within the data. This is likely due to the very few users who achieved a higher score of 7 or 8 as discussed above.

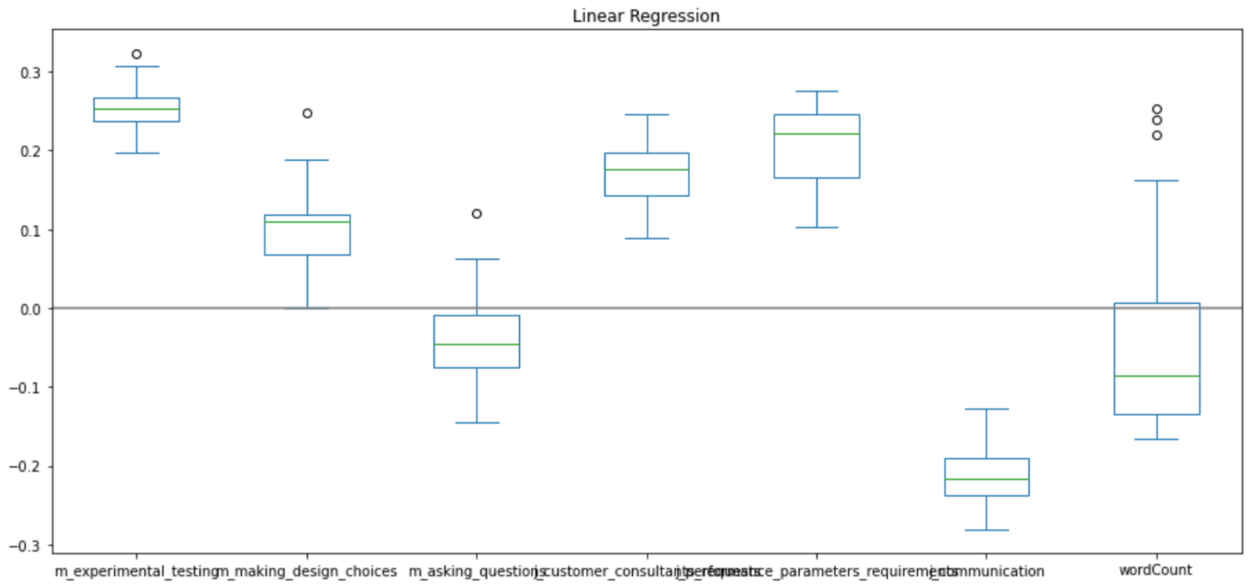
Modelling

Overview

The main goal of each of these models was to predict a final outcome score as accurately as possible. Two data sets were used in these predictions, the dataset grouped by individual users, in which we tried to predict individuals outcome scores, and the dataset grouped by teams in which we were attempting to predict the average score per user in each of these groups.

Linear Regression

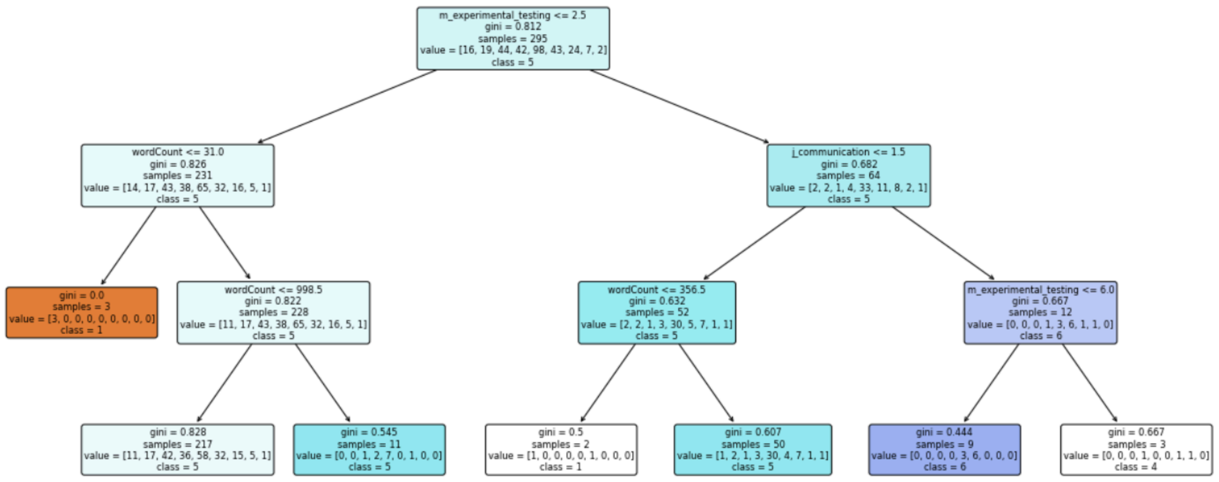
The first model utilised was a simple Linear Regression model as it is easy to implement and gave an initial baseline for performance. This however returned very low training scores of 0.108 and 0.230 and testing scores of -0.164 and -0.102 for the grouped by user and by team data respectively. This negative testing score means that our data doesn't follow this model at all. A potential cause of issue in Linear Regression models is often the high amount of variance within the data which causes the algorithm to model this erroneous data and not the intended outcome. This can be seen graphed below for the user model.



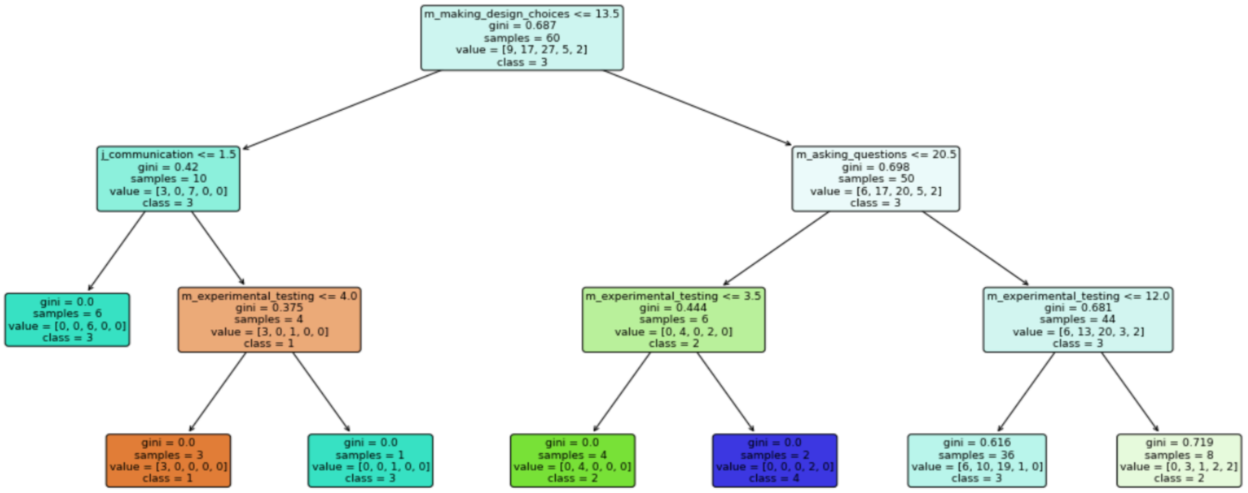
This variance can also lead to overfitting, meaning that the model is far too specific to the training data provided. We look to minimise the amount of variance in this model without increasing too much the bias in what is known as the Bias-Variance Tradeoff. Regularisation is designed to help with this by simplifying the data and there are two types which were utilised, Lasso and Ridge Regularisation. However, even with these techniques implemented there was only a very slight improvement of testing score, but it still remained negative and therefore an unfit model for the data.

Decision Trees

The next model implemented was a decision trees classifier model. This was deemed to be an appropriate model to use as each of the user scores was a discrete target value we were attempting to find. However, this did require the use of rounded versions of the average user score per group in order to make them discrete and provide easier comparison between the two models. Shown here is the user model with a depth of 3.


m_experimental_testing <= 2.5
gini = 0.812
samples = 295
value = [16, 19, 44, 42, 98, 43, 24, 7, 2]
class = 5
wordCount <= 31.0
gini = 0.826
samples = 231
value = [14, 17, 43, 38, 65, 32, 16, 5, 1]
class = 5
i_communication <= 1.5
gini = 0.682
samples = 64
value = [2, 2, 1, 4, 33, 11, 8, 2, 1]
class = 5
gini = 0.0
samples = 3
value = [3, 0, 0, 0, 0, 0, 0, 0, 0]
class = 1
wordCount <= 998.5
gini = 0.822
samples = 228
value = [11, 17, 43, 38, 65, 32, 16, 5, 1]
class = 5
wordCount <= 356.5
gini = 0.632
samples = 52
value = [2, 2, 1, 3, 30, 5, 7, 1, 1]
class = 5
m_experimental_testing <= 6.0
gini = 0.667
samples = 12
value = [0, 0, 0, 1, 3, 6, 1, 1, 0]
class = 6
gini = 0.828
samples = 217
value = [11, 17, 42, 36, 58, 32, 15, 5, 1]
class = 5
gini = 0.545
samples = 11
value = [0, 0, 1, 2, 7, 0, 1, 0, 0]
class = 5
gini = 0.5
samples = 2
value = [1, 0, 0, 0, 0, 1, 0, 0, 0]
class = 1
gini = 0.607
samples = 50
value = [1, 2, 1, 3, 30, 4, 7, 1, 1]
class = 5
gini = 0.444
samples = 9
value = [0, 0, 0, 0, 3, 6, 0, 0, 0]
class = 6
gini = 0.667
samples = 3
value = [0, 0, 0, 1, 0, 0, 1, 1, 0]
class = 4

From this it can be seen the different requirements the model takes to classify input factors into different final outcome scores. For both the user and group based models, through manual testing and the gridsearchcv function, an optimal depth of 8 was found for both models. This provided enough depth to accurately classify scores while also being not too deep within the tree as after a certain point the model becomes highly specified to the provided data resulting in a very large bias. Ultimately, the user model obtained an accuracy of 0.311 and the group model, pictured below, had a final accuracy of 0.333.

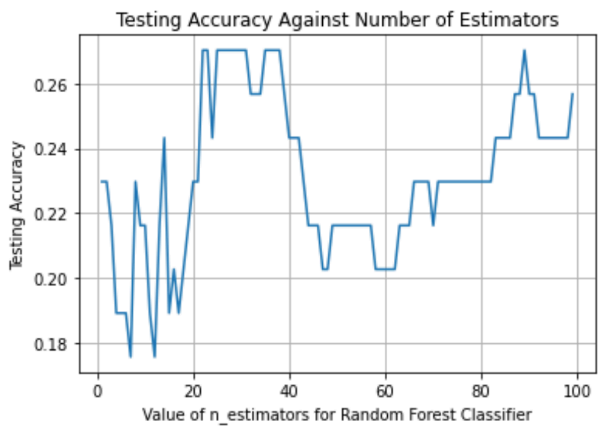


Both of these models are improvements upon the initial linear regression, however still leave much to be desired.

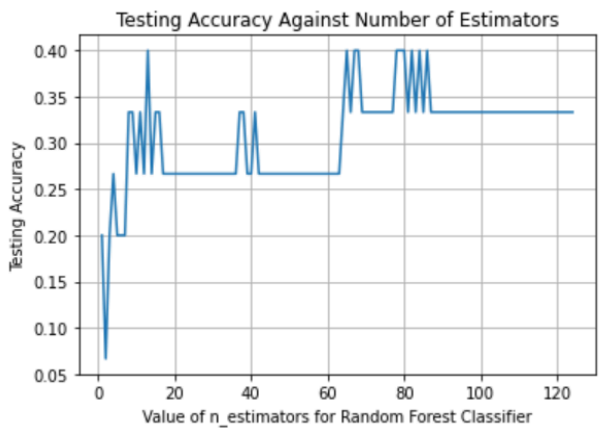
Random Forests

Following on from the decision tree modelling, we moved on to a Random Forest Classifier model. This was selected as it is an aggregation of multiple Decision Trees meaning that ideally, using these multiple models, it will provide us with a more accurate final model. Once again the rounded average scores were used for the group model as discrete values are required for classification as with Decision Trees. In order to find the best accuracy for the models we needed to calculate the optimal amount of estimators for the Random Forest Classifier to use. More estimators tends to result in a higher accuracy however it also makes it into a slower model. Below can be seen the graphs of number of estimators against model accuracy for both the user and group models.

User Model



Group Model



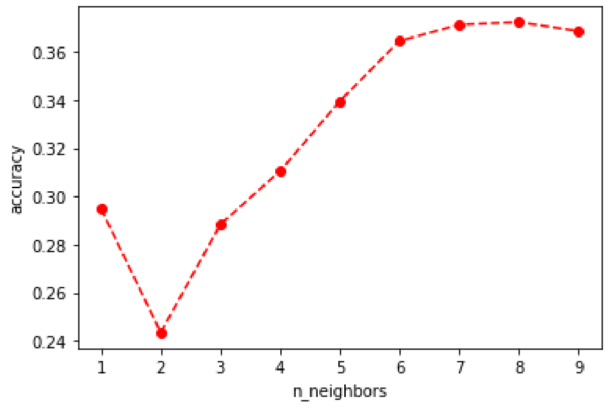
Through the use of these visualisations in conjunction with gridsearchcv the optimal amount of estimators was found to be 30 and 100, for the user and group models respectively, which return accuracies of 0.270 and 0.333. These amounts of estimators were chosen as they were when the accuracy was highest but also had stopped varying as much, as seen in the plots.

However, the accuracy scores obtained from this model showed no improvement whatsoever over the original Decision Tree model results with the user model actually decreasing in accuracy by 0.041 while the group model remained the same. While this result is unexpected, it is likely due to the fact that Random Forest models are a collected average of multiple Decision Trees. In the Decision Tree models we may have just been lucky to find an accurate random state, while on average the accuracy score of other Decision Tree models is much lower.

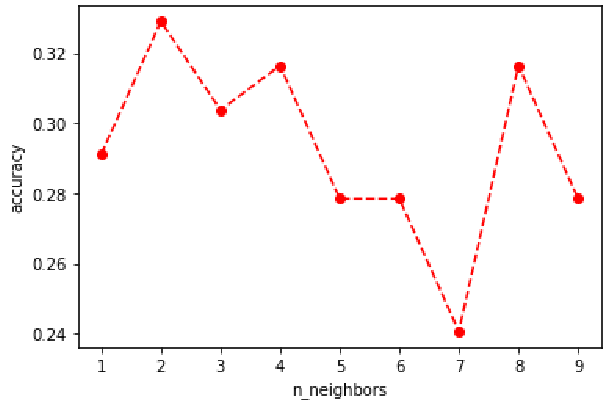
Another benefit of Random forests is that it can tell you which features are the most important for the model. For the user model 'word count', 'asking questions' and 'making design choices' were deemed the most important whereas for the group model 'experimental testing' replaced 'word count.'

K-Nearest Neighbour

- Group Model:



- User Model:

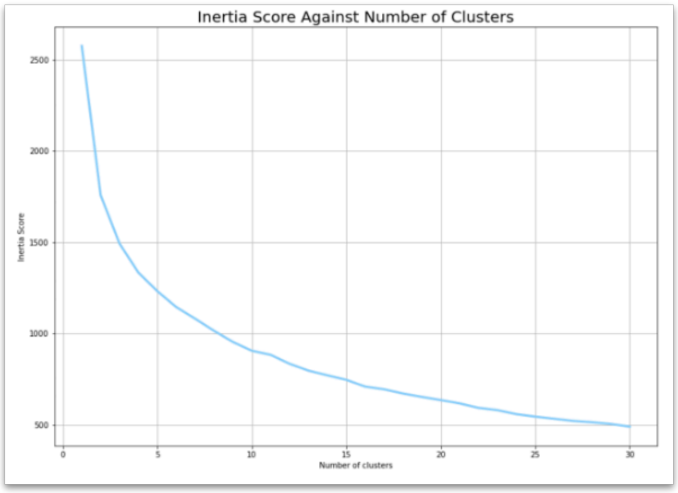


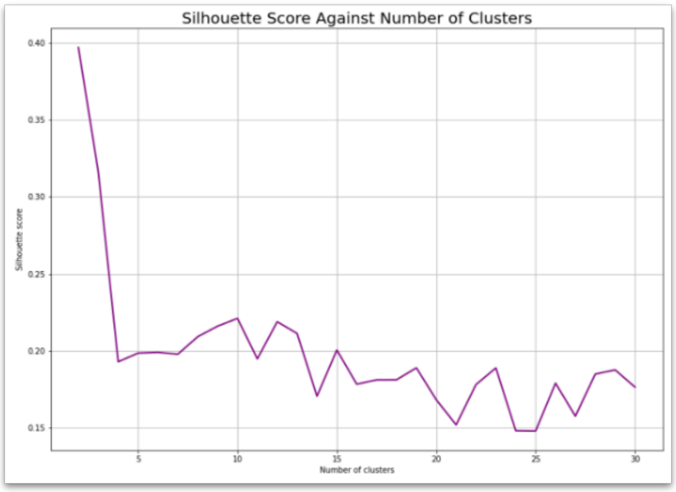
We implemented the K-Nearest Neighbour classification model. The K-Nearest Neighbour (KNN) algorithm is a data classification method used to estimate the likelihood that a data point is a member of one group or another based on which group the nearest data point belongs to (Joby, 2021).
To optimise our model, we varied the number neighbours parameter from 1 to 9 and evaluated the performance of each setting on both models. For each 'n_neighbors' value, we trained the KNN classifier and then used it to predict the outcome of the test dataset. Afterwards, we calculated the accuracy of each prediction, which measures the proportion of correct predictions out of the total number of instances. The results are displayed in a line graph, allowing us to see how the accuracy of the model varies as the number of neighbours grows.
The highest accuracy score we achieved was approximately 0.38 in the group model. For the user model, we achieved 0.28. While this score reflects the model's ability to correctly predict outcomes based on the given training data, its relatively low accuracy score suggests that this may be due to the fact that the words or phrases extracted from the chat logs do not adequately reflect the knowledge or skills that translate into reported performance, as well as if the chat data contains a lot of irrelevant or misleading information. KNN is a simple, distance_based classification method (Aishearya Singh, 2018). If the relationship between chat content and reported scores is complex or nonlinear, KNN may not be the best model for this particular dataset.

KMeans Clustering

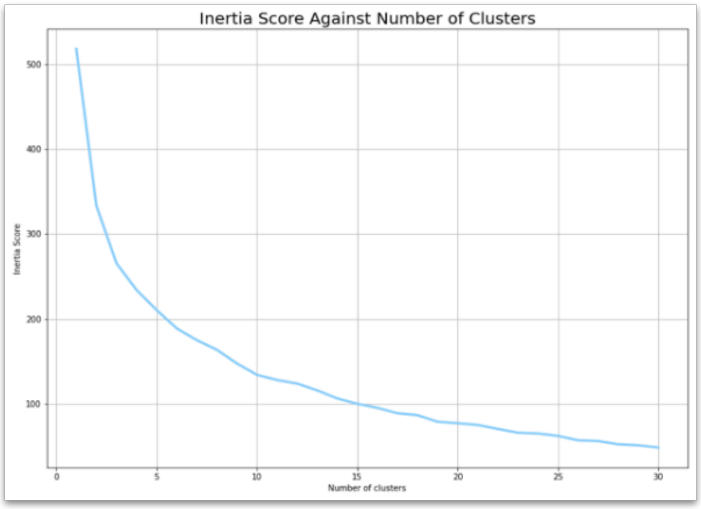
K-means clustering is utilised to identify potential structures or groups in our dataset that may not be initially apparent(LEDU, 2018). Clustering could reveal patterns related to how different team behaviours or chat patterns correlate with their final Outcomescores. We initially set the algorithm to k=3, which implies the data will be separated into three clusters. Each cluster will contain data points that are similar to one another in terms of the clustering features. The fitting approach is used to compute k-means clusters, while the prediction method is used to compute cluster centres and predict the clustering index for each data point.
The kmeans.inertia_ parameter is used to assess the quality of our clusters. This value represents the total of the samples' squared distances to their nearest cluster centres. A lower inertia score indicates better clustering in this scenario since it indicates that the data points are closer to their respective centroids.

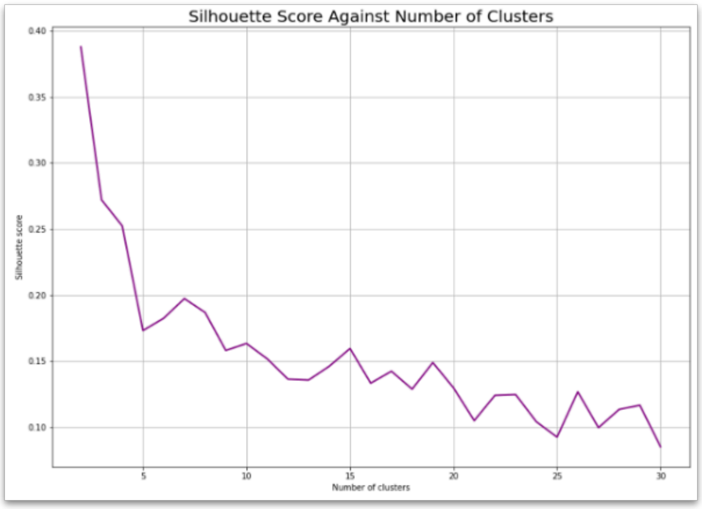
- User Model:





- Group Model:

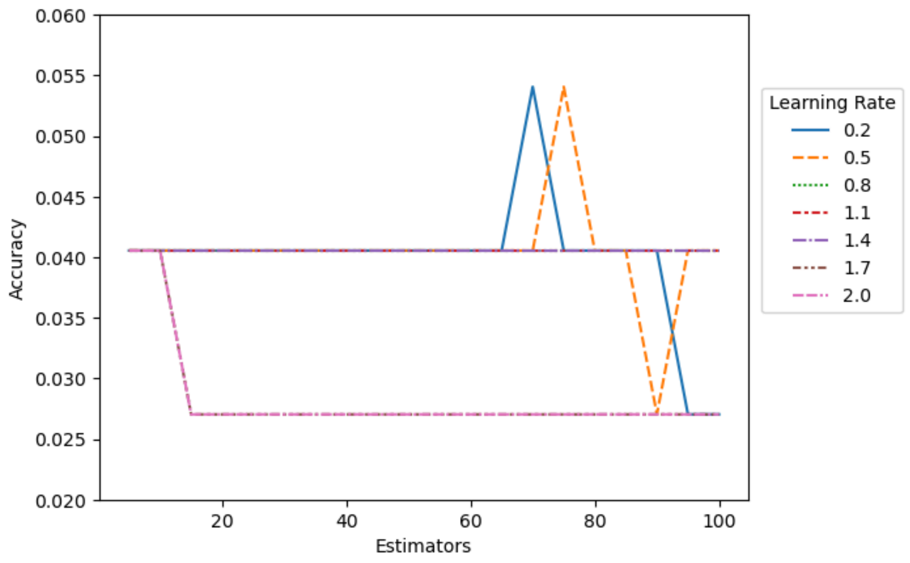




From those images,we can see that it loops over a range of possible cluster numbers (from 1 to 30), fitting a k-means model for each number of clusters and storing the resulting inertia score. In the User model, we set that n_clusters=10 when initialising the KMeans model to have an optimal clustering solution. 905 is the inertia score , indicating the total within-cluster sum of squares. In the Group Model, we set n_clusters=7 and got 175 inertia scores.

Boosting

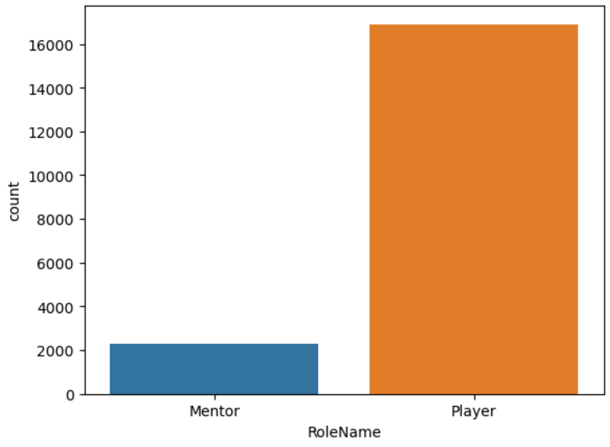
Boosting, similar to kMeans clustering is an iterative algorithm, this means an estimate is created, then used to generate an updated and theoretically improved estimate, this algorithm then continues until a stopping criterion is satisfied, or it reaches a maximum number of iterations. The boosting implemented with our data uses simple and fast regression and classifications models at each iteration to create the model. In particular AdaBoost (Adaptive Boosting) was used which is an iterative process that associates weights at each observation, the weights are then shifted after each iteration, so more emphasis is placed on classifications that are incorrectly predicted, these weighted targets then create the updated model. Learning rates the rate at which the weights are updated each iteration and consequently choosing an optimal learning rate is vital when optimising AdaBoost, a learning rate too high or too low decreases the accuracy of the model.

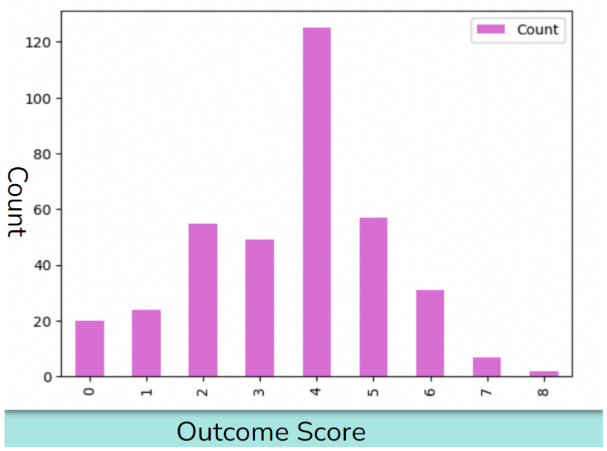


The graph above represents the optimal number of estimators and learning rate to achieve the highest accuracy, for the data provided the best learning rate is 0.5 and the corresponding optimal number of estimators is 75.

SMOTE

Imbalanced data occurs when observed frequencies are clearly different across different values of a variable, in other words there are many of one variable type and few of another, two examples of this can be seen below with majority of the users being players over mentors and a third of the users having an outcome score of 4, and an uneven spread of scores from 1-8.





It is important to combat this imbalance as a model that predicts an outcome score of 4 may have a high accuracy, however this is irrelevant to our analysis due to the number of outcome scores of 4. One method to counter data imbalance is under sampling which is when a number of data points that are present too often are removed from the data, the disadvantage to this is that valuable data is lost. Oversampling on the other hand consists of making duplicates of the data values that are least present in the data, this however leads to the creation of many duplicate data points. Data augmentation is a method that works like oversampling, however rather than creating duplicate data points, minor deviations are made to the copied data points, the Synthetic Minority Oversampling Technique (SMOTE) uses data augmentation to derive an advanced version of oversampling, thus avoiding duplicates of data. The aim of the SMOTE algorithm is to increase false positives and decrease false negatives, however it does this at the cost of accuracy, as the accuracies of our models are already so low this is not a desirable approach when predicting outcome scores.

NLP

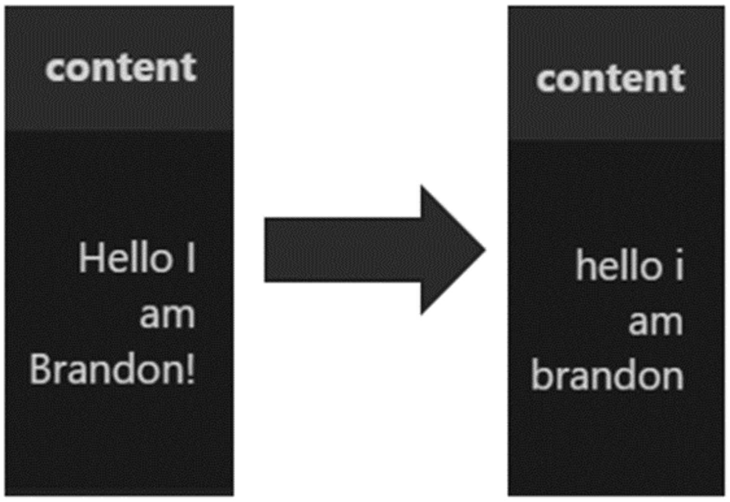
Overview

Natural Language Processing (NLP) is a field of programming concerned with extracting usable numeric data or other important information from human language or text.

When it comes to the Nephrotex Virtual Internships dataset, the actual chat or 'content' column is the primary source of information which determined other features such as 'm_making_design_choices' or 'j_communication', which are Boolean columns which denote the purpose or function of a chat message. As such, to introduce any new features or information to the dataset, they would have to be extracted from the actual chat transcripts using NLP techniques.

For this dataset, TF-IDF will be used to find the most important words in all the chat data and use them to establish new features which may have some stronger and more relevant correlations to the outcome score.

Text-Data Pre-Processing



Before we apply any NLP techniques to the data, the text data needs to undergo some transformation to make the TF-IDF more effective. Most NLP techniques, including the one intended to be used on this dataset, involve tokenizing the text by words or splitting bodies of text by individual words. However, features of language things such as capital letters or standard features of punctuation, mean that the tokens 'Mean', 'mean.' and 'mean' are recognized as different words. To avoid any inaccuracies, the chat data was stripped of all punctuation and set to lower case only.

TF-IDF

Term frequency-inverse document frequency (TF-IDF) is an NLP technique used to find the importance of each word in a document or series of documents, in relation to those documents. This is achieved by finding the number of times a certain term appears in one document (the term frequency) and multiplying it by the proportion of documents in the set which contain that (inverse document frequency), this process usually returns a score between 0 and 1, where the higher the score the more relevant, or important, the word can be considered. Whilst the TF helps to recognize words with high usage, the IDF helps balance this frequency count by returning lower scores for those words which appear in more documents, to avoid confusion caused by standard lexical features of the English language, e.g., ‘the’, ‘is’, ‘I’, etc.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

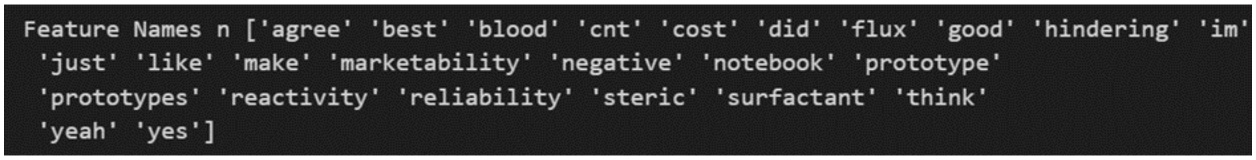
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

The formula used to calculate TF-IDF

Implementing TF-IDF on the Virtual Internships data involved utilising the scikit learn package ‘TfidfVectorizer’. When initialising the vectorizer, stop-words were set to ‘English’ so that basic structural words in the English language would be ignored by the vectorizer. The vectorizer was also initialised to return a list of 25 words with highest TF-IDF scores, using the ‘max_features’ parameter, this was aimed to make initial analysis a bit easier and prevent over-analysis of the TF-IDF results in-order to avoid confusion.

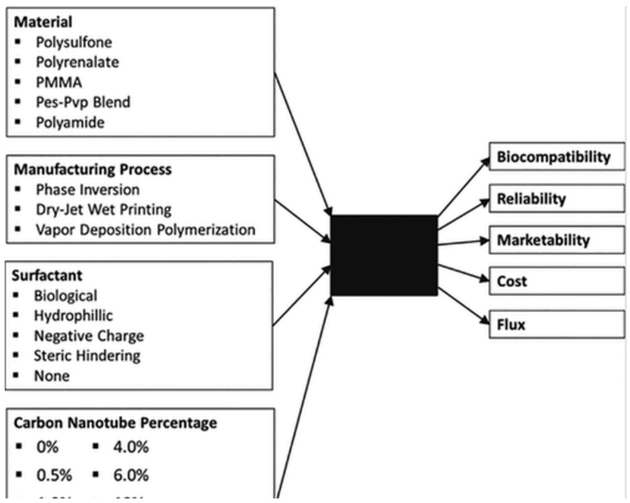


```
Feature Names n ['agree' 'best' 'blood' 'cnt' 'cost' 'did' 'flux' 'good' 'hindering' 'im'
 'just' 'like' 'make' 'marketability' 'negative' 'notebook' 'prototype'
 'prototypes' 'reactivity' 'reliability' 'steric' 'surfactant' 'think'
 'yeah' 'yes']
```

List of the 25 most important words according to the TfidfVectorizer.

New Features

Though there are now 25 possible words from which new features could be designed it is important to note that the issue of relevancy had to be considered. Making features without any reason would cause inaccuracies in any later modelling and analyses.



A diagram describing the inputs and outputs considered by participants in the Virtual Internship program.

Upon some further research about Nephrotex, it was learned that some components of the virtual internship task could be categorised as either inputs or outputs. Using the diagram seen to the side and the list extracted from the TfidfVectorizer it was possible to construct two new, contextually relevant, features for the dataset. Both of which counted the following words in each chat message, producing a sum of the number of input and output terms contained in one chat entry. The hypothesis being the more a participant used input and output terms the more likely they were having relevant conversations which would lead to higher outcome scores.

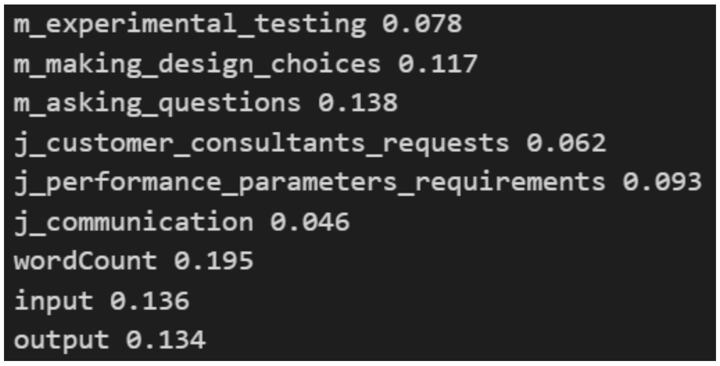
Input:

- Surfactant
- Steric
- Hindering
- CNT (Carbon Nanotubes)

Output:

- Reliability
- Cost
- Flux
- Marketability

Modelling with New Features

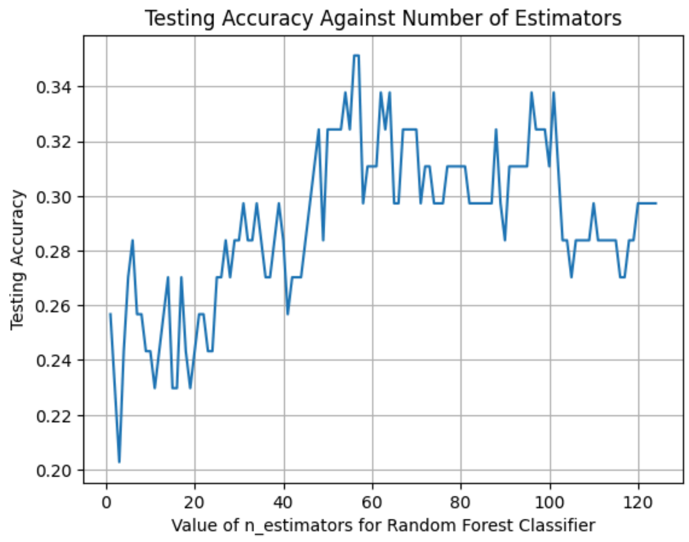


```
m_experimental_testing 0.078
m_making_design_choices 0.117
m_asking_questions 0.138
j_customer_consultants_requests 0.062
j_performance_parameters_requirements 0.093
j_communication 0.046
wordCount 0.195
input 0.136
output 0.134
```

To begin exploring the effects of the new features, a user-based model was constructed using Random Forest Classifier. This model produced a 0.297 score for accuracy at an optimal number of estimators of 69, which was an improvement from the previous user model's 0.27, however it was not substantial enough to be deemed relevant.

Though there was no evident improvement in accuracy due to addition of new features, when taking a look at the feature importances for the model, both 'input' and 'output' are the most important features after word count.

This was an indication that creating the new features were a step in the right direction when it came to constructing better models.



Taking the feature importances into consideration a new model using RFC was made using just 'output', 'input', 'wordCount', 'm_asking_questions' and 'm_making_design_choices', the features with importance greater than 0.1.

This model was capable of achieving an accuracy of 0.351 when the number of estimators was 55, however as shown in the graph the accuracy is very volatile and lacks stability around any number of estimators.

Results

Initially, a Linear Regression model returned a training and testing score of 0.108 and 0.164 for the user model and 0.230 and -0.102 for the group model. An attempt was made to reduce the variance within the data using Lasso and Ridge regularisation however this only slightly improved the scores with testing score still being negative indicating an unfit model. Following this, a Decision Tree Classifier was utilised which returned accuracies of 0.311 and 0.333 at a depth of 8 for the user and group models respectively. To improve upon this a Random Forests Classifier was employed however this returned a decreased accuracy score of 0.270 with 30 estimators for the user model, and an accuracy of 0.333 with 100 estimators for the group model. The Random Forests user model was further improved when using the new features, ‘input’ and ‘output’, whilst also removing some less important features to generate an accuracy of 0.351 with 55 estimators.

Both a K Nearest Neighbours (KNN) and kMeans clustering model was developed to analyse the data, the accuracy for the KNN model was 0.28 and 0.38 for the user and group model respectively, this group model was the highest accuracy achieved thus far. The kMeans clustering model doesn’t use training and testing sets and instead only uses the data features to fit the model, it also doesn’t calculate an accuracy score and instead determines the optimal number of clusters and the inertia score. The optimal number of clusters for the user and group models was 10 and 7 respectively and the user model obtained an inertia score of 905 while the group model obtained an inertia score of 175, the inertia score represents how well a dataset was clustered, and a good model has a low inertia score and a low number of clusters and thus the group model was the better model in both situations.

Conclusion

Various models were applied to the virtual internships data in both a user and group sense, the highest accuracy score achieved was 0.38 in the group model in K Nearest Neighbours (KNN) with the next closest being a Random Forests Classifier produced using Natural Language Processing which returned an accuracy of 0.351. The approaches used to predict outcome score using chat data between groups faced limitations as the different topics discussed and the frequency of communication was shown to have little to no effect on the outcome score, groups with below average scores were still talking just as much and about the relevant things. A third of the users in the internship also achieved an outcome score of 4 and the outcome score is on a user level based on the report the individual submits, this made it difficult to predict a group outcome score as groups had to be given a score derived from the members averages. Natural Language Processing was utilised to provide an in depth analysis of the chat data between groups, specifically TF-IDF (Term frequency - Inverse document frequency) was used to determine the importance of certain words, however this method faced various limitations such as its lack of account for sentiment/meaning and instead is just frequency based, the words also had to have meaning assigned by us and we could be wrong. The next step in natural language processing would be conducting sentiment analysis to provide meaning for words and demonstrate a direct relation in the words to performance outcome. Mentors were also removed from the modelling as they were all given an outcome score of 4 and it would bias the results, in future implementations of the Nephrotex internship it would be beneficial to have the same set of mentors in each internship as it would provide opportunity to analyse the effect of mentors on the outcome score.

Appendix

References

- **Nephrotex - International Society for Quantitative Ethnography**.
  https://www.virtualinterns.org/nephrotex/

- **Aishwarya Sigh. (2018, August 22). *KNN Algorithm: Introduction to K-Nearest Neighbours Algorithm in Regression*. Medium.**
  https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/
- **Swalin Alvira. (2018, Jan 31). *How to Handle Missing Data.* Towards Data Science. Medium.**
  https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4
- **Education Ecosystem(LEDU). (2018, Sep 13). *Understanding K-means Clustering in Machine Learning.* Medium.**
  https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
- **Shaffer, David & Arastoopour, Golnaz & Swiecki, Zachari & Ruis, Andrew & Chesler, Naomi. (2015). Teaching and Assessing Engineering Design Thinking with Virtual Internships and Epistemic Network Analysis.**
- **Golnaz Arastoopour, Chesler, N., D'Angelo, C. M., & Lepak, C. (2012). Nephrotex: Measuring first year students' ways of professional thinking in a virtual internship. ResearchGate; unknown.**
  https://www.researchgate.net/publication/283865973_Nephrotex_Measuring_first_year_students'_ways_of_professional_thinking_in_a_virtual_internship
- **Korstanje, J. (2021, August 30). SMOTE. Medium.**
  https://towardsdatascience.com/smote-fdce2f605729

Figures

Fig 1.

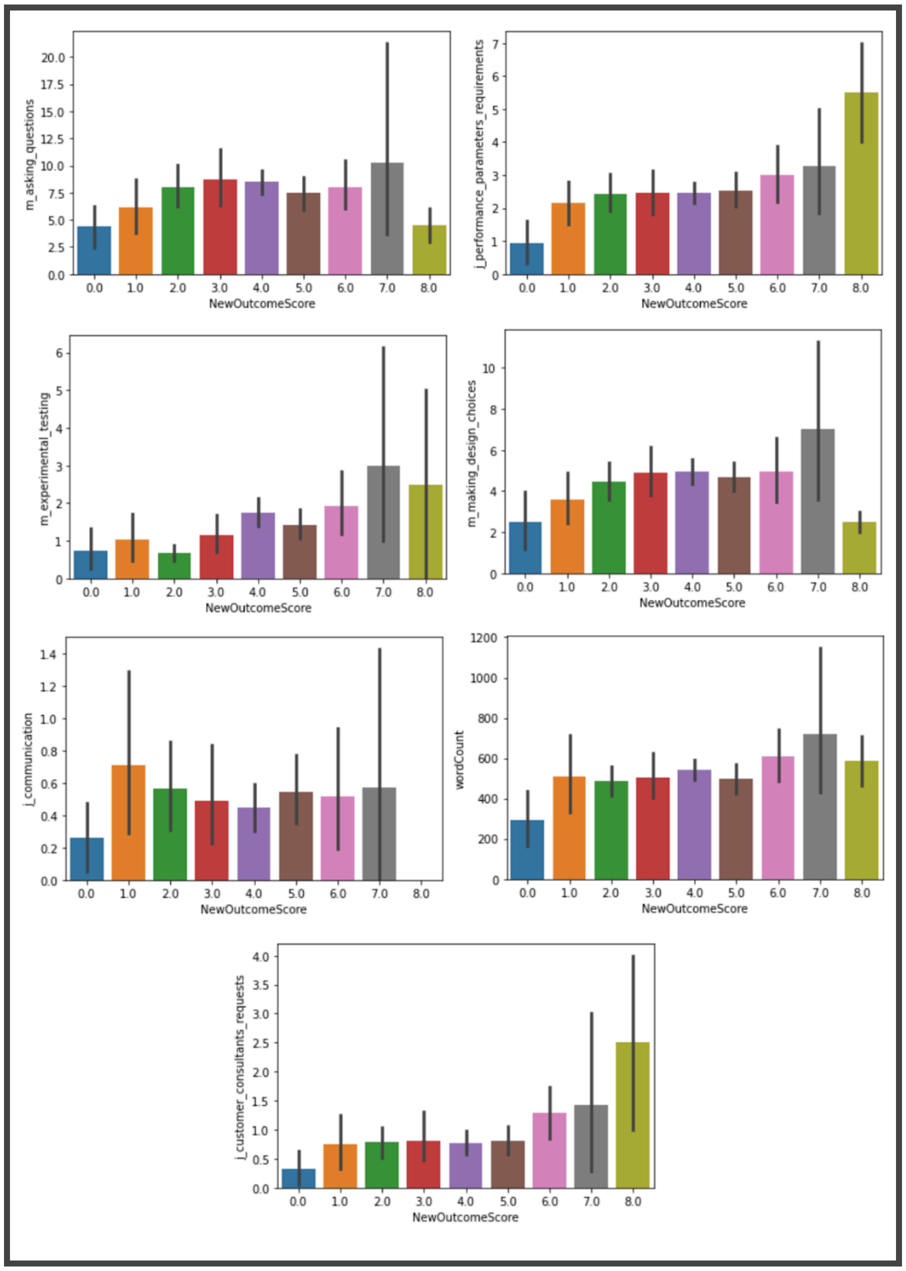
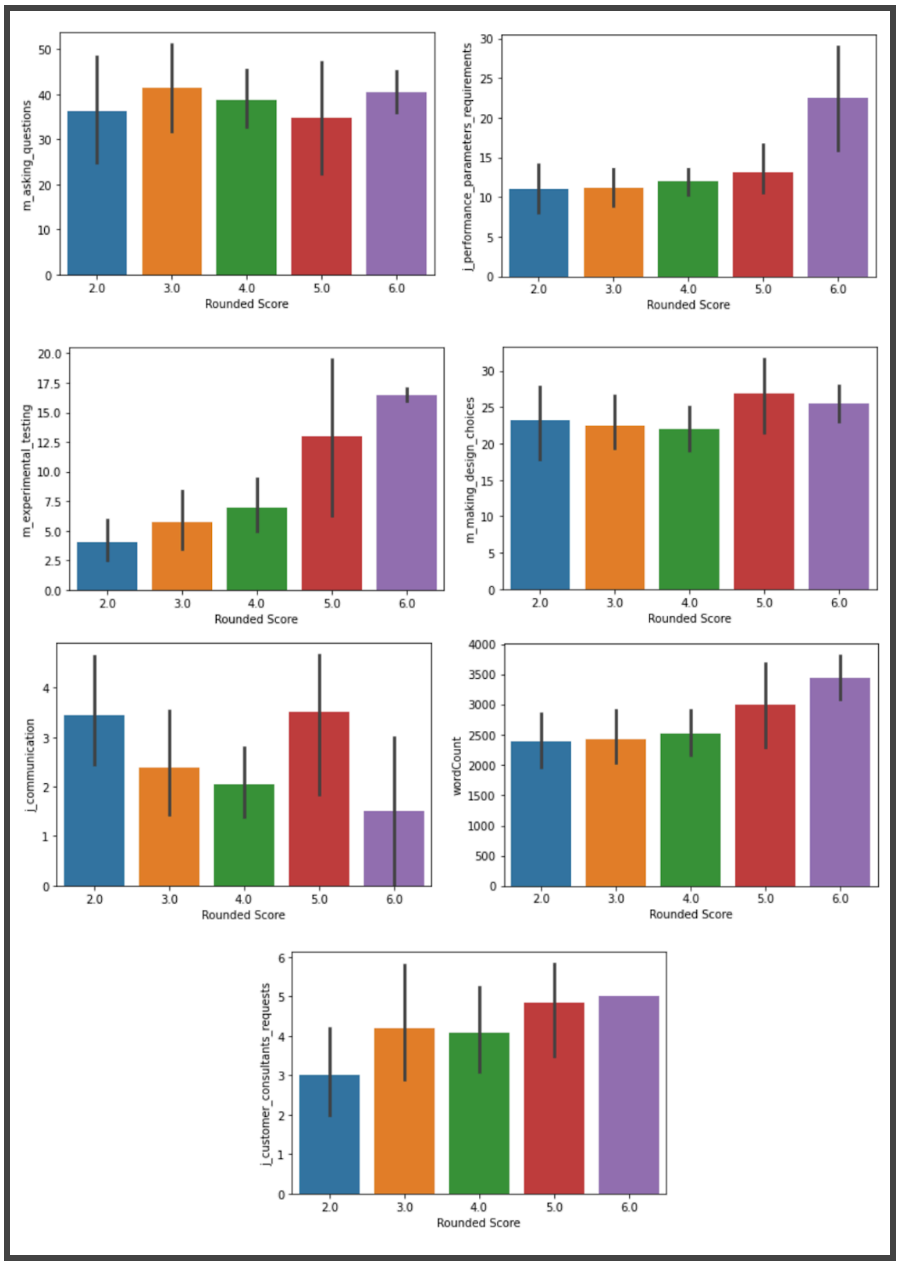

m_asking_questions
j_performance_parameters_requirements
m_experimental_testing
m_making_design_choices
j_communication
wordCount
j_customer_consultants_requests
NewOutcomeScore
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0

Fig 2.


m_asking_questions
j_performance_parameters_requirements
m_experimental_testing
m_making_design_choices
j_communication
wordCount
j_customer_consultants_requests
Rounded Score
2.0
3.0
4.0
5.0
6.0