Final Project - Meta Cognition

Gal Waisman and Sergiy Horef

October 8, 2024

Code for the project - GitHub

Contents

1 Dataset

Corresponds to points 1,2 in the requirements.

1.1 Data Collection

We have chosen to scrape Reddit for posts in the following subreddits:
r/: macapps, learnprogramming, learntodraw, learnpython, learnmath, LaTeX, Python, datascience, dataengineering, malefashionadvice, MachineLearning, ObsidianMD, neuroscience, printSF, science, ios, MacOS and mac.
We have chosen these subreddits, as we have noticed that they have a high amount of "question posts" (posts that have one or more correct answers).

We have scraped the data using python, and the following main libraries: 'selenium' and 'BeautifulSoup' (the code can be found in the github repository provided).
We have filtered the posts by their title, and only included the ones which have either "?", "question" or "help" in the title text.
Because reddit has a tree-like structure of comments - that is, each comment can have subcomments, etc. we have only included the comments which were made to the post itself, and therefore can be the possible answers.

For each comment we have collected the following information:
1. Score - difference between the likes and dislikes.
2. Replies - number of replies (sub-comments) for that comment.
3. Awards - number of awards (if any) that were given to the comment.
4. Length - number of symbols in the comment.
5. Length to Average Ratio - ratio between the length of a given comment to the average comment on that post. ($\frac{Length}{avg.Length}$)

We have not collected the text of the question or the answer itself, as this data would be much harder for the model to use, and would take much more time for the people to read.
Moreover, we have felt that the five numbers we have collected will be enough for the model to meaningfully train.

In total we have succeeded in collecting 1274 comments, out of which we have chosen 50 that would be used for model training and human labeling (level of sureness).
We have chosen these comments such that they would be "interesting" and include relatively high scores, length and reply numbers with the following statistics:

stat/value	score	replies	length	length ratio
min	-4	0	5	05.49%
mean	93.74	2.36	118.74	64.78%
max	1437	37	744	288.62%
std	224.17	5.75	133.57	54.67%

We have chosen not to include the "awards" value for each of the comments, as it has shown to be non-informative, with most values being a 0, and only a few being 1 or 2.

1.2 Data Labeling

In order to label the 50 coments we have chosen, we have split them into 2 groups of 25 each, and each was labeled by 5 different people.

Each person got a personal explanation of what the data represents, and how it was collected. Later, each one was shown an excel-like table, with each row representing an individual comment, and each column filled with the corresponding value of that measure ("score", "number of comments", "length", "length ratio").
Last column was empty, and represented the sureness (0-100%) of the comment writer as estimated by the person. Each person was asked to fill this column based on their subjective feeling and understanding.

Later, each comment got its final label to be the average of 5 labels provided by the people in its group. We have wanted to make the final labels an average of a few opinions, as otherwise intra-person bias could lead to an incoherent data.

The final labels follow the following statistics:

stat	value
min	58.2
mean	76.8
max	92
std	8.87

Or in a box plot:

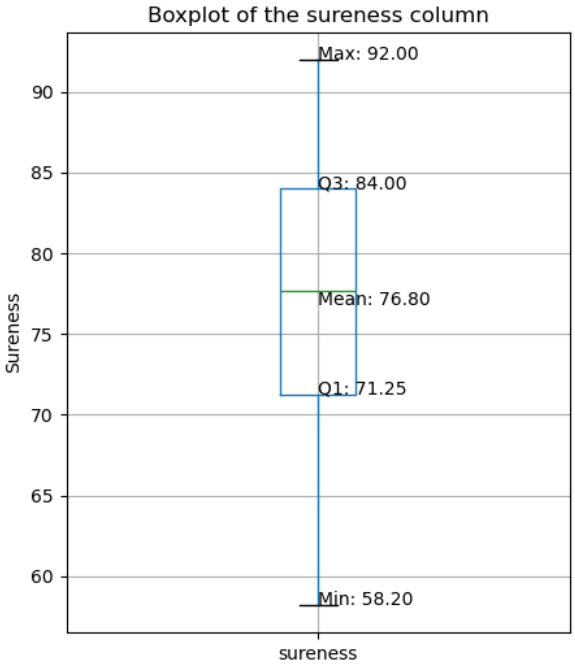


Figure 1: Statistics of the labeled data

2 Models and Results

Training of all models can be found in 'main.py'
Corresponds to point 3 in the requirements.

We have not collected either the questions themselves, or the full text of the comments, therefore we could only use models that do not rely on natural human language.

Models: we have tried 5 different models: 1 baseline model, 3 simple ones of the "fit-predict" sort, and a simple neural network.
Each model was trained to predict a real number in range 0-100 to be as close to the "real" label (as explained in the previous section). In order to make sure that predicted values were valid, prediction of each model had an additional step which clipped all of the values to the 0-100 range. We talk more about each model in the following subsections.

Evaluation: for each of the models we have done 5-fold cross validation, where the data was initially split into 5 (disjoint) sets of data points, with 10 points each. Later, each fold was used for testing, while other folds were unified and used for training of the model.

For each test, we have measured the mean l1 - mean absolute error (MAE), and mean l2 - mean squared error (MSE), between the predicted labels, and the "real" ones.

Note: in all models that have used any sort of randomness, and for the division into splits in cross-validation, we have used the seed of 3.

2.1 Baseline

Implementation can be found in 'Baseline.py'

From labels distribution box plot can be seen that even though the total range of the data is relatively big, ranging from 58 to 92 (92-58=34), most of the data is concentrated around the mean value of 76.8 (Q3-Q1=84-71.25=12.75 or one-sided: Q3-Mean=84-76.8=7.2).
Therefore, in order to check the real usefulness of other models, we have created a baseline model - when trained, it saves the mean value of the training labels, and on the test stage it simply always returns that value.

This model achieves **MSE of 79.27** and **MAE of 7.39**.

2.2 K-Nearest Neighbors (KNN)

Implementation can be found in 'KNearestNeighbors.py'

We have started by creating a weighted regression KNN model, implementation for which we have taken from the sklearn module. (Standard module for machine learning tasks.) Given a point, this model finds k nearest points in the training set, and returns a label which is the weighted (by l2 distance) average of the k points.

We have tested all values of k from 1 and up to 40 (always average all points), and found that the best result is achieved at $k = 6$ with **MSE of 55.2** and **MAE of 5.89**. That is, on average, this model returns labels that are 5.89 away (to either side) from the real value. Not bad!

Here is the plot of performance as a function of different values of k:

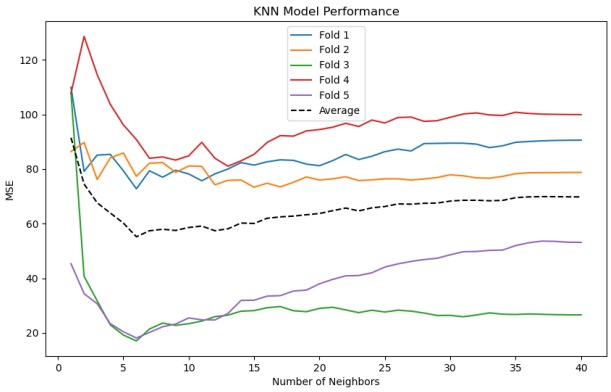


Figure 2: Graph of KNN performance

2.3 Decision Tree

Implementation can be found in 'DecisionTree.py'

Next we have implemented a Regression Decision Tree (using sklearn).
Given a set of training points, this model creates a decision tree, where at each step it splits the data into two groups by value of one of the parameters. When a new point is classified, it moves through the tree until it reaches a leaf group, and receives a value which is the average of all values in that group.

We have tested all depths from $d = 1$ and up to $d = 20$, and found that the best result is achieved at $d = 1$ with **MSE of 91.91** and **MAE of 7.74**. The tree for this depth looks like this:

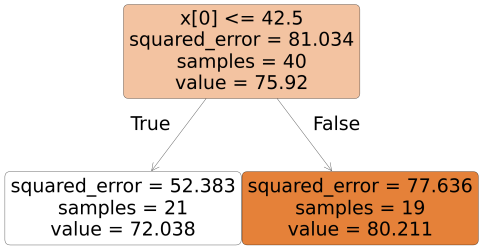


Figure 3: Decision Tree of depth 1 (trained on folds 1-4)

That is: the average over all labels is 75.92, and the split is done using the 'score' parameter. Anything that has a score of 42.5 or less, will be assigned a label of 72.038, and else a label of 80.211.

The overall performance of the Decision Tree model as a function of the depth:

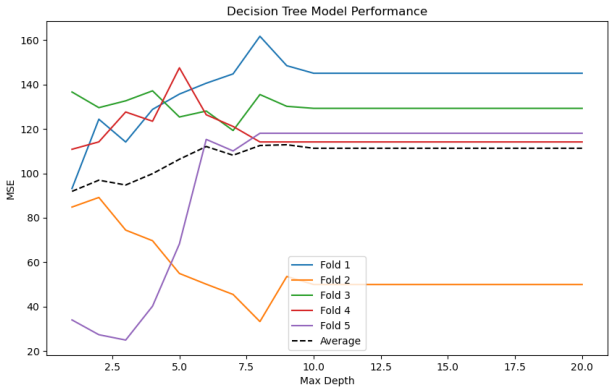


Figure 4: Performance of Decision Tree on various folds

2.4 Feed Forward Neural Network

Implementation can be found in 'FeedForwardNetwork.py'

As we have only 40 points for each fold, training a deep learning model is impractical. As almost all of them are only relevant when the number of points is relatively high, or if the data is very homogeneous.
However we still wanted to see how a deep learning framework would perform, and therefore chose to train a simplest form of a feed forward neural network.

We have tested different numbers of layers from $n = 1$ and up to $n = 5$, and found that the best results were achieved over $n = 3$ (size of the hidden layer is 9), with **MSE of 95.32** and **MAE of 8.11**. Much worse than the simpler models, but rather as expected.

Here is an example of the MSE and loss of the model on one of the folds:

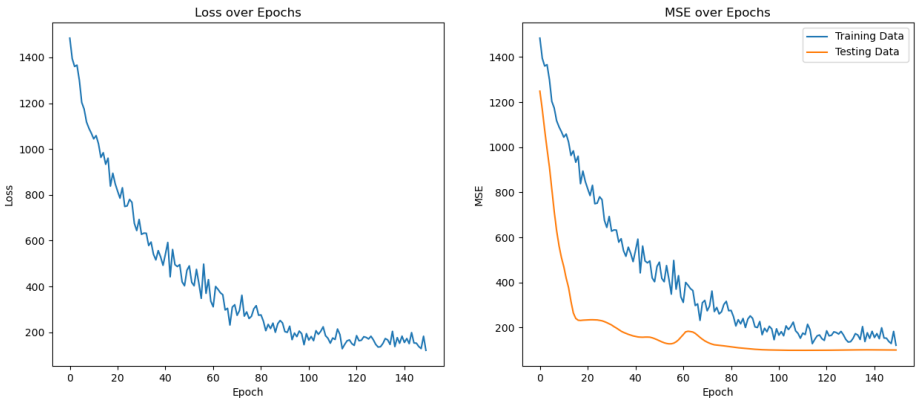


Figure 5: Performance of an FFN on one of the folds

We can clearly see that after around 90 epochs of training the model converges to its best result. (Even though this result is not really good.)

2.5 Linear Regression

We have used the LinaerRegression module implemented in sklearn.

As our last model, we wanted to train a Linear Regression model, to try and understand which factors (possibly) had the most effect on the label assigned by our friends. Here are the results for each fold, with coefficient values, MSE and MAE:

fold/val	score	replies	length	length ratio	MSE	MAE
fold 1	0.0070	-0.8193	0.0047	-1.1185	82.62	7.62
fold 2	0.0073	-0.4781	0.0086	-2.0259	73.43	7.46
fold 3	0.0072	-0.6053	0.0072	-3.9961	50.07	6.67
fold 4	0.0078	-0.4738	0.0038	-1.3141	89.85	8.24
fold 5	0.1689	-0.4822	0.0171	-2.4431	107.49	8.52

Table 1: Coefficients and results of Linear Regression on different folds

And on average, the results were **MSE of 80.69** and **MAE of 7.70**. This result is still much worse than what was achieved using the simple KNN, but is better than either Decision Tree or FFN.

2.6 Intermediate Conclusions

First of all, lets look at a general comparison between all of our models (the best values (less is better) are highlighted in bold):

Model	MSE	MAE
Baseline	79.27	7.39
KNN	**55.2**	**5.89**
Decision Tree	91.91	7.74
FNN	95.32	8.11
Linear Regression	80.69	7.70

Table 2: Comparison of model performance

It can be clearly seen that only the KNN model can be said to have learned something, as all others perform worse than even the baseline, which does no computations whatsoever.
Linear Regression provides results that are close to Baseline, but are still a little worse. Neural Network performs as expected given such a small training set - produces results that are much worse than any other model, and almost a whole unit farther than Baseline in the MAE value.

In light of all of the results, we can state a few conclusions:

- The good performance of the KNN model suggests that the heuristics our friends have used in order to label each example were conditional on a few similar, or previously labeled, examples. (Recall that each person was asked to label 25 points in total, and the best k value for KNN was 6.)
  This seems to agree with the theory that most of human judgements are comparison based - there is no some single baseline, but rather everything is defined in relation to something else. And also with the properties of human short term memory, which can only hold a few pieces of information at a time.

- Poor performance of the Decision Tree model seems to suggest that whatever rule was used in creation of the labels, it cannot be satisfactorily explained by rigid 'yes-no' splits. However, if we did try to find a best possible explanation of this sort - it would include only a single split by the score parameter.

- Poor performance of the neural network is very much expected. Deep learning models are very complex, and therefore require either a very uniform training set, where the rule is very simple and obvious; or a relatively big training set, such that the model would be able to capture the complexity of the data. (And our data is quite complex, as we explain in the following points.)

Poor perfomance of the Linear Regression model requires a few points to unpack.

- Straight away we see that the performance of the linear regressor is worse than the null model (baseline - always return average) which immediately rings that bell of something is not right. As theoretically, such a thing should never happen. There might be a few possible reasons.

- First of all, we want to point out that linear regression models reguire certain assumptions to work properly (linearity, homoscedasticity, independence and normality). We know for sure that at least the independence assumption doesn't hold in our case as each part of 25 points has biases coming from 5 different people. It is quite possible that other assumptions are not true too.
- Secondly, due to the nature of the internet and the data we have collected, there are inherent correlations between different cues (see Correlation Heatmap). Here are possibly some of them:
  - There is a highly clear positive between the length and length_ratio (simply due to the fact that length_ratio is derived from length.)
  - There is a negative correlation between the length/length_ratio and score, due to the fact that people (on Reddit) usually tend to only read shorter comments. Therefore shorter comments achieve a greater amount of likes.
  - There is a slight positive correlation between the replies and length_ratio, which can lead to a hypothesis that longer comments tend to attract longer and more in-depth replies.
- There are also some inherent factors in the nature of the internet forums themselves which might lead to data points which "dirty" the data:
  - There are "trolls" and "flamers" who are two types of people whose sole purpose is to write questions/comments which lead to a high positive or negative engagement.

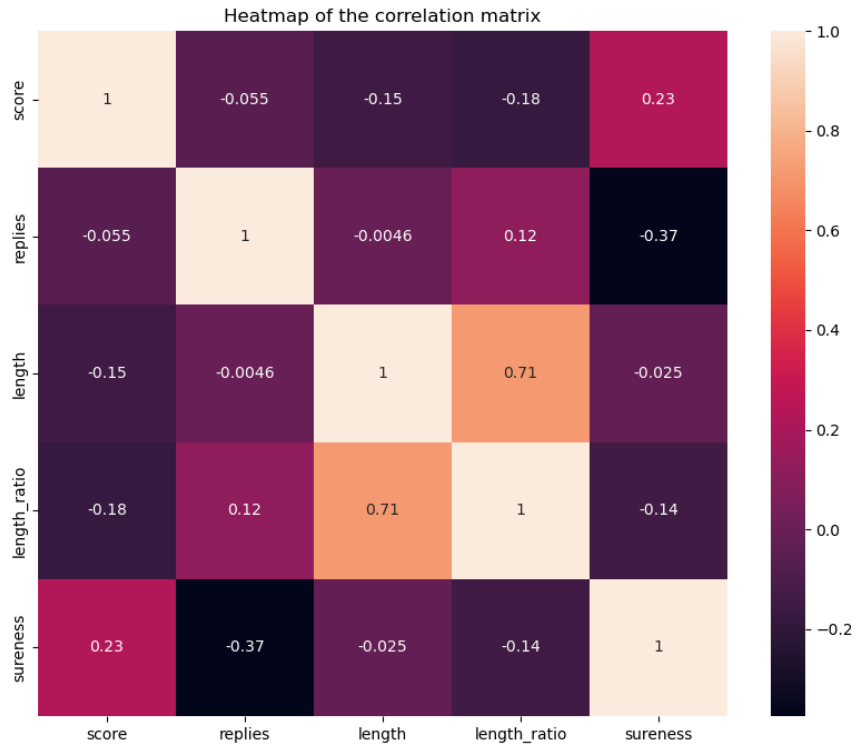


Figure 6: Heatmap of correlation between different parameters (cues) of the data

3 Explainability

Corresponds to points 4&5 in the requirements.

3.1 Problem Definition

Explainability is the problem of different (usually were complex) machine learning models where the model works in a "black box" way. That is, the model learns a connection between the input and output features, however, it is unclear how this inference is made.

Some of the accepted ways of reducing the level of unexplainability include:
1. Visualizations of the output vectors provided by the model at each stage of its training. This helps in uderstanding how the clusters of models decisions change through time.
2. Training simpler and explainable (e.g. Decision Trees) models that learn to approx-

imate the behaviour of the compelex and unexplainable one. Looking at the decision rules of the simpler model may help to understand the behavior of the complex one.
3. Feature Importance and Counterfactual methods are used to understand which features influence the models output the most, and conversely, small changes in which features lead to the biggest change in the model's prediction. This might help to glimpse the hidded connections between input and output features learned by the model.
4. Prototype and Criticism Based Methods are used to find which examples are thought by the model to be most representative of each class, and which are the most overlooked ones. This might help to understand the semantic representations learned by the model.

3.2 Metacognition in Explainability

In tasks of sureness prediction, metacognition may be crucial to understaning how the labels for each data-point were created.

Because machine learning models are trained to infer the label given the data, they are learning to approximate the connection between them.
Understanding metacognition may help us to understand how the humans have produced the labels, and therefore the possible connection that the model may have found. Furthermore, understanding the process underlying label creation in humans may help us to notice biases present in the training set, which would help to explain the biases expressed in the final model.

3.3 Properties that help Explainability

As we have said earlier, in our data we have only 4 possible cues to the sureness, and we regard each of them below:

- Score:
- Replies:
- Length & Length Ratio: both of these cues may be similar to the cue of time it took to solve a problem we have seen in class.

4 Use of BEVoCI Methodology

Corresponds to points 6&7 in the requirements.

First of all, we want to point out that in our case there is no "examinee".
Linear Mixed effects models (which are the basis for the Hierarchical Linear Regression, which are in terms the basis for nlme library in R and therefore for BEVoCI) account for the possiblity of dependent data points, which are a result of the same person responding to different questions (or same questions across time). They do so by introducing a

group (or person)-based random effects.

In our case, and due to the fact that our data is a combination of 5 different people (otherwise we would not be able to train a deep learning model), "examinee" approach would not make sense.

4.1 BEVoCI for Explainability of Neural Network

If we looked solely on the performance on each of the models, we would have chosen to work with the KNN model, and we are not sure (conceptually) how the use of BEVoCI methodology would help in this case. Therefore, and due to the fact that we believe that applying BEVoCI to Neural Networks was the purpose of this project, we have chosen to do the BEVoCI analysis of the Feed Forward Neural Network predictions.

In order to use the BEVoCI model to try and understand the cues that our neural network (Feed Forward Neural Network) has used to make its predictions, we have trained the model on all of the available points (50), and made predictions on all of the same points. We have than saved this data into a new file (ffn_data) upon which we have run Hirarchical Regression.

Contrary to the R language, python does not have a native implementation of the lme (Linear Mixed Effects) models. Therefore (instead of implementing everything by hand), we have found a user-made implementation (HLR python library), and we express a deep gratitude to its creator. (Link to the github repository with the code - link.)
We have selectively checked the code to make sure that it does what it is supposed to, and the creator himself writes that he checked many of the results against a code produced by a specialized library in SPSS.

In addition to the standard implementation of the Hirarchical Linear Regression, this library also calculates the F value, R^2, partial correlations, MSE and unique variance explained by each additional variable.
This library also includes the following statistical tests for each individual regression:

- **Independence of residuals (Durbin-Watson test)** - test for autocorrelation in the residuals from a statistical model or regression analysis. The Durbin-Watson statistic will always have a value ranging between 0 and 4. A value of 2.0 indicates there is no autocorrelation detected in the sample. Values from 0 to less than 2 point to positive autocorrelation, and values from 2 to 4 mean negative autocorrelation. (Source)

- **Linearity (Pearson r)** - a correlation coefficient that measures linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations. (Source)

- **Linearity (Rainbow test)** - even if the true relationship is non-linear, a good linear fit can be achieved on a subsample in the "middle" of the data. The null hypothesis is rejected whenever the overall fit is significantly worse than the fit for the subsample. (Source)
- **Homoscedasticity (Breusch-Pagan test)** - tests whether the variance of the errors from a regression is dependent on the values of the independent variables (in this case there is heteroskedasticity). (Source)
- **Homoscedasticity (Goldfeld-Quandt test)** - used to test the presence of Heteroscedasticity in the given data. It does so by splitting the data into two sets with respect to a single chosen variable, and compares the residuals in the two sets. (Can miss heteroskedasticity if the split is done w.r.t a variable that does not have heteroskedasticity.) (Source 1, Source 2, Source 3)
- **Multicollinearity (pairwise correlations)** - checks for a situation where (some of) the predictors in a regression model are linearly dependent. (But only pariwise correlations.)
- **Multicollinearity (Variance Inflation Factors)** - measures the multicollinearity between the predictors by looking at the ratio (quotient) of the variance of a parameter estimate when fitting a full model that includes other parameters to the variance of the parameter estimate if the model is fit with only the parameter on its own. (Source)
- **Outliers (extreme standardized residuals)** - checks for unusual outliers by looking at the extremely high or low (higher than 3 or lower than -3) values of the residuals. (Source)
- **Outliers (high Cooks distance)** - tries to detect outliers by looking at the Cooks distance. It is a measure of the effect of deleting an observation on the estimated coefficients. It takes into account both the leverage and the residual of the observation. High Cooks distance indicates that the observation has a large effect on the estimated coefficients when its deleted. (Source)
- **Normality (mean of residuals)** - checks for the assumption of normality of residuals by looking at their mean. (Normality assumption holds if the mean is around 0.)
- **Normality (Shapiro-Wilk test)** - specific numerical test to check the residual normality assumption. (Source)

Many of these tests were unknown to us previous to encountering them in this library, so we cannot say that we fully understand either the tests themselves, or their implications; but we have chosen to include their results in any case for a more knowledged reader to interpret.

Additionally, we would like to point out that we have not encountered the concept of Hierarchical Regression Models before this projects, therefore we in no way claim deep understanding and/or proficiency in the use of these models. Therefore, some of our conclusions might be invalid or simply wrong, however, this is due to inexperience but not malice.

We have tried the following hierarchies:
(In each model the dependent variable is 'sureness' and we list the independent variables we have used for the regression.)

Hierarchy 1

- score
- score, replies
- score, replies, length
- score, replies, length, length ratio

Hierarchy 2

- score
- score, replies
- score, replies, length ratio

Hierarchy 3

- score
- score, length

Hierarchy 4

- score
- score, length ratio

Hierarchy 5

- replies
- replies, length

Hierarchy 6

- replies
- replies, length ratio

Hierarchy 7

- length
- length, length ratio

Hierarchy 8

- replies
- replies, score
- replies, score, length ratio

In all of the hierarchies (except for the first one, just to see what would happen) we have included either the length or the length ratio, but never both. They have a very high correlation, and therefore do not pass the rule of thumb presented in the BEVoCI paper, which is to include a cue only of correlation is less than 0.3. For that matter, models which include both cues also do not pass the multicollinearity tests.

By looking at the statistical results over for all of the hierarchies, we can come to a few conclusions:

- No combination of cues is valid enough to use Linear Regression at all. Plausibly, this is due to the fact that many of the tests checking for the assumptions of linear regression fail for most of the cues.
- In general, it seems that the informative content (higher R^2 and lower p-value for validness of regression) is ordered as follows: score > replies > length > length ratio

Therefore, we present here the statistical results only for the most informative model/hierarchy - hierarchy 1 with the cues: score, replies, length. However, the complete results (including all of the corresponding graphs) can be found in the 'data_exploration.ipynb' file, or in the pdf export which we include with this file.

Here is a table of (selected) statistical results for hierarchy 1:

cues	R^2	F-value	p-value (F)
score	0.0562	2.8582	0.0974
score, replies	0.101	2.6282	0.0828
score, replies, length	0.1276	2.2428	0.096

Table 3: Statistical Results for Hiearachy 1 - part 1

cues	coefs (const; cues)	p-value coefs (const; cues)
score	76.36; -0.0037	0; 0.0974
score, replies	76.68; -0.0039; -0.1295	0; 0.08; 0.13
score, replies, length	77.24; -0.0043; -0.13; -0.0044	0; 0.056; 0.13; 0.24

Table 4: Statistical Results for Hiearachy 1 - part 2

cues	semi-partial corrs.	R^2 change	F-value change	p-value (F change)
score	-0.24	-	-	-
score, replies	-0.25; -0.21	0.044	2.32	0.13
score, replies, length	-0.27; -0.21; -0.16	0.027	1.42	0.24

Table 5: Statistical Results for Hiearachy 1 - part 3

Here are the results from the statistical tests (in the same order in which we have presented them earlier - Statistical Tests) (test result, p-value (if relevant)):

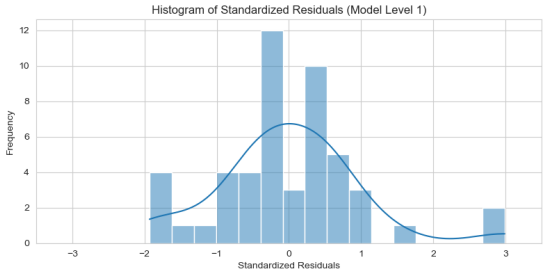
cues	DW	Pearson r	Rainbow	BP	GQ	Pairwise
score	T	(F, 0.097)	(T, 0.39)	(T, 0.64)	(T, 0.91)	T
score, replies	T	=; (F, 0.17)	(T, 0.25)	(T, 0.67)	(T, 0.94)	T
score, replies, length	T	=; =; (F, 0.39)	(T, 0.25)	(T, 0.47)	(T, 0.94)	T

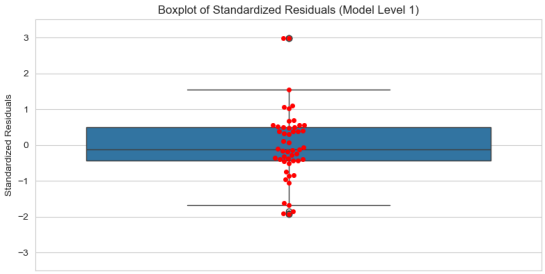
Table 6: Statistical Tests for Hierarchy 1 - part 1

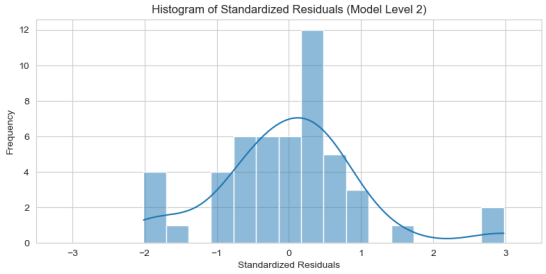
cues	VIF	Extreme Residuals	Cooks	Mean Residuals	SW
score	T	T	T	T	(F, 0.006)
score, replies	T	T	T	T	(F, 0.004)
score, replies, length	T	T	T	T	(F, 0.009)

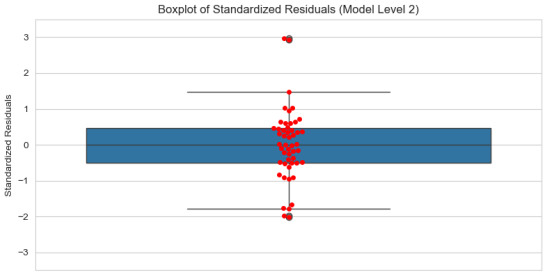
Table 7: Statistical Tests for Hierarchy 1 - part 2

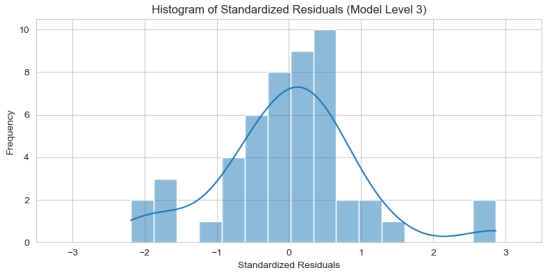
Plots of the graphs of different statistical values:

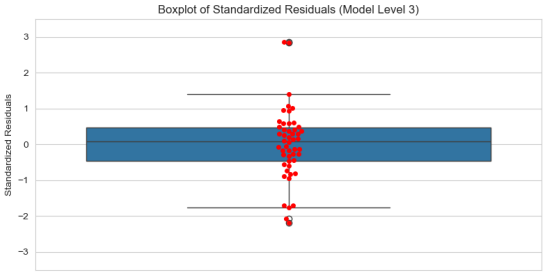


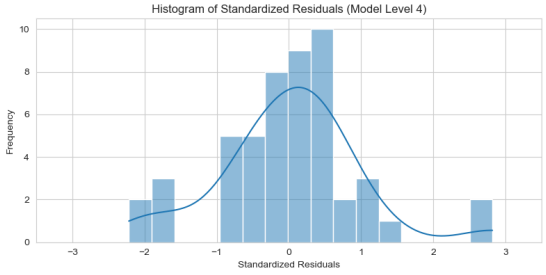


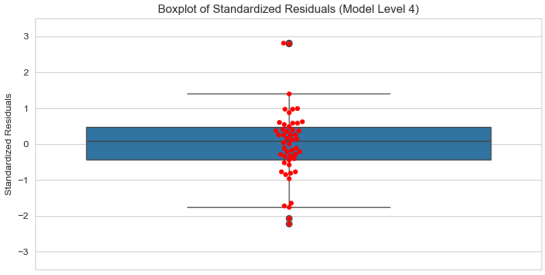




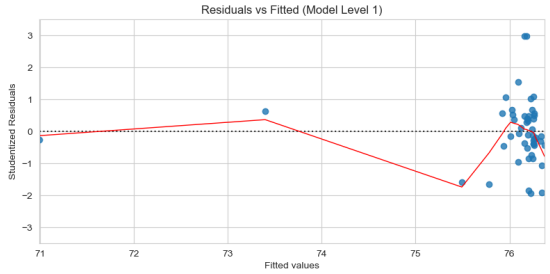


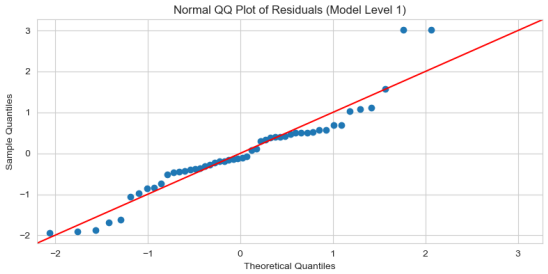


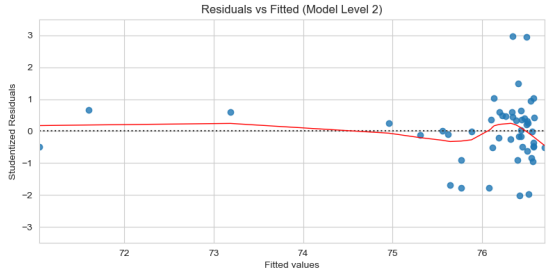


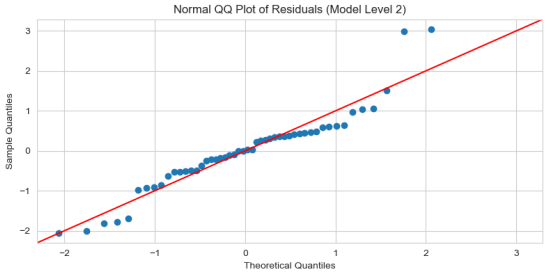


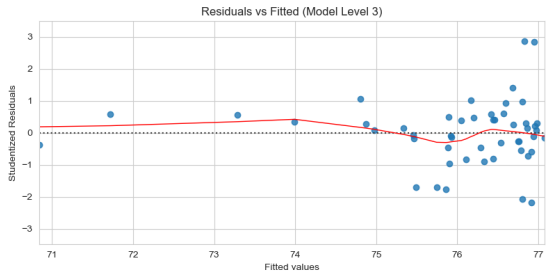
(a) Standardized Residuals Historgram (b) Boxplot of Standardized Residuals

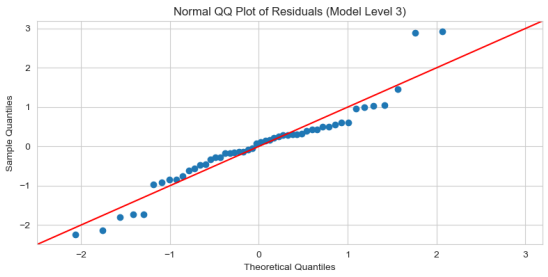


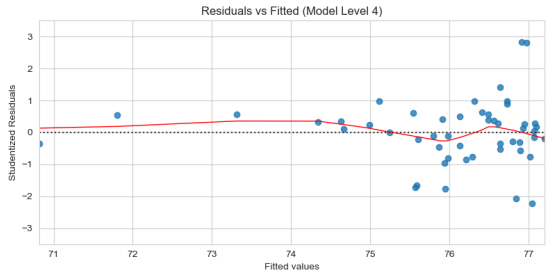


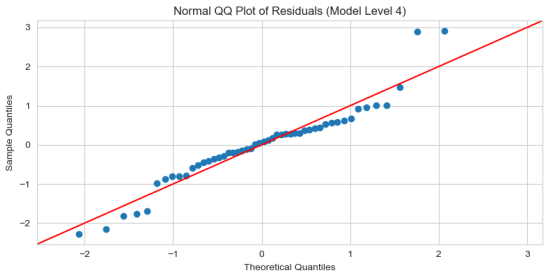












(a) Standardized Residuals plot

(b) QQ plot of Standardized Residuals

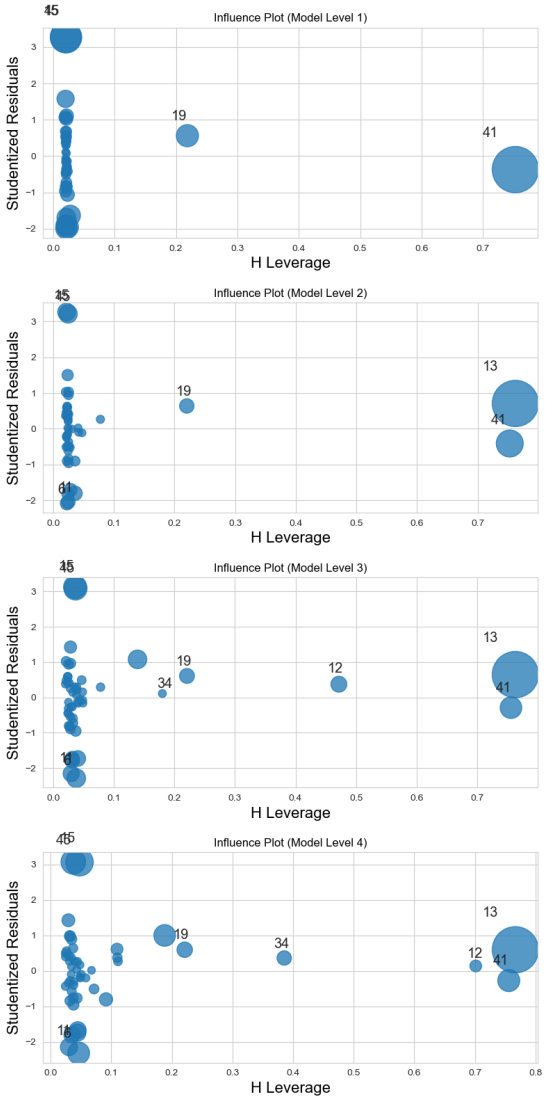
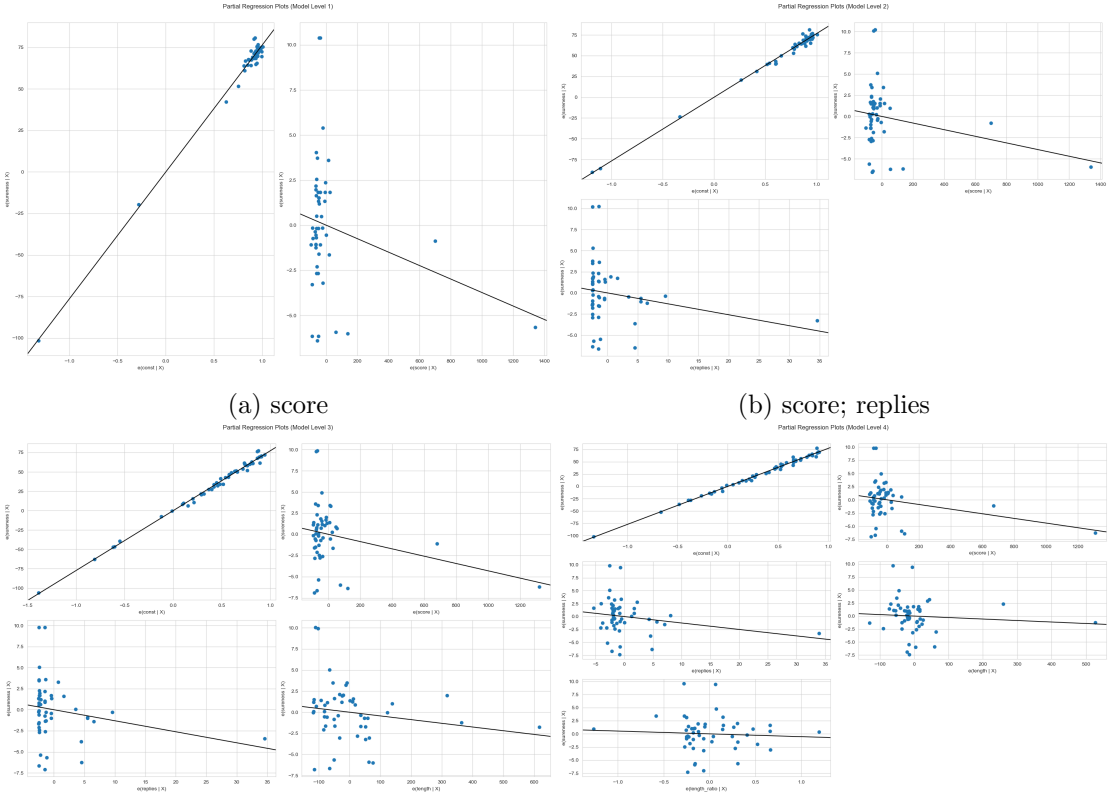


Figure 9: Plot of influence of Different Points on the model



(a) score

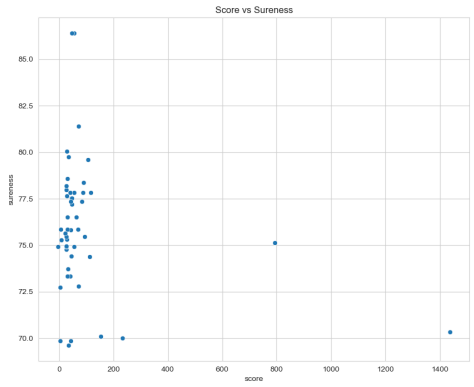
(b) score; replies

(c) score; replies; length

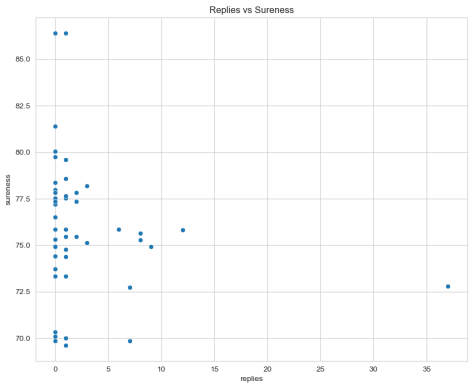
(d) score; replies; length; length ratio

Figure 10: Plots of Standardized Residuals as a function of each individual cue + constant

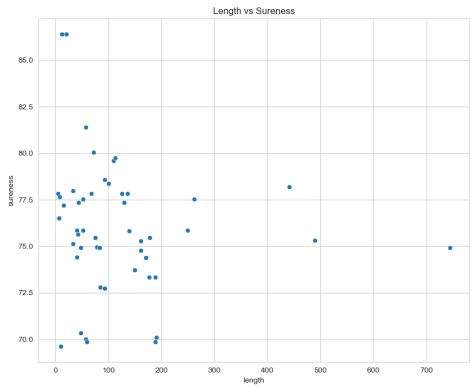
Plots of different cues and their effects on the final sureness:



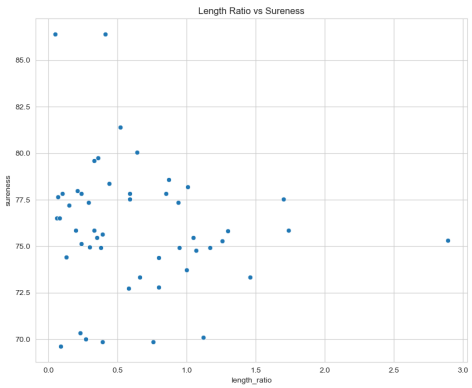
(a) Score - Sureness



(b) Replies - Sureness



(c) Length - Sureness



(d) Length Ratio - Sureness

Figure 11: Different Cues and their Effects on the final Sureness

4.2 BEVoCI for Meta Cognition

After applying the BEVoCI methods to the data produced by the Feed Forward Neural Network, we were interested in applying the same method to the origianal data - this would help us understand if the performance of the neural network was so bad due to the small number of points, or rather due to the general lack of regularity in the data.

In a similar way to the analysis of the Feed Forward Network, here too, we come to the following conclusions:

- There is only one cue which makes use of Linear Regression valid - replies. (Other cues do not get this validity, plausibly due, again, to the fact that many of the tests checking for the assumptions of linear regression fail for them.)
- In general, it seems that the informative content (higher R2 and lower p-value for validness of regression) is ordered as follows: replies > score > length ratio >

length. (Notice that it is quite different from the results we got in BEVoCI for Meta Cognition.)

Therefore, we present here the statistical results only for the most informative model/hierarchy - hierarchy 8 with the cues: replies, score, length ratio. However, the complete results (including all of the corresponding graphs) can be found in the 'data_exploration.ipynb' file, or in the pdf export which we include with this file.

Here is a table of (selected) statistical results for hierarchy 1:

cues	R^2	F-value	p-value (F)
score	0.0562	2.8582	0.0974
score, replies	0.101	2.6282	0.0828
score, replies, length	0.1276	2.2428	0.096

Table 8: Statistical Results for Hiearachy 1 - part 1

cues	coefs (const; cues)	p-value coefs (const; cues)
score	76.36; -0.0037	0; 0.0974
score, replies	76.68; -0.0039; -0.1295	0; 0.08; 0.13
score, replies, length	77.24; -0.0043; -0.13; -0.0044	0; 0.056; 0.13; 0.24

Table 9: Statistical Results for Hiearachy 1 - part 2

cues	semi-partial corrs.	R^2 change	F-value change	p-value (F change)
score	-0.24	-	-	-
score, replies	-0.25; -0.21	0.044	2.32	0.13
score, replies, length	-0.27; -0.21; -0.16	0.027	1.42	0.24

Table 10: Statistical Results for Hiearachy 1 - part 3

Here are the results from the statistical tests (in the same order in which we have presented them earlier - Statistical Tests) (test result, p-value (if relevant)):

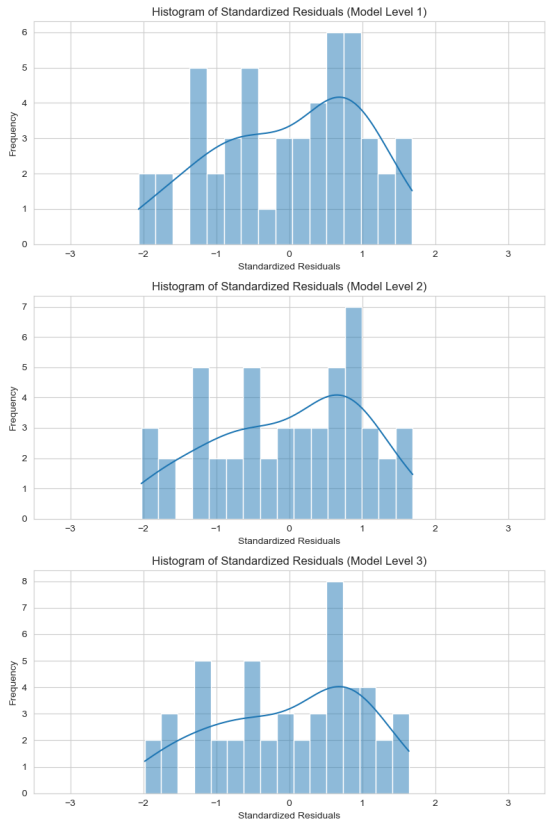
cues	DW	Pearson r	Rainbow	BP	GQ	Pairwise
score	T	(F, 0.097)	(T, 0.39)	(T, 0.64)	(T, 0.91)	T
score, replies	T	=; (F, 0.17)	(T, 0.25)	(T, 0.67)	(T, 0.94)	T
score, replies, length	T	=; =; (F, 0.39)	(T, 0.25)	(T, 0.47)	(T, 0.94)	T

Table 11: Statistical Tests for Hierarchy 1 - part 1

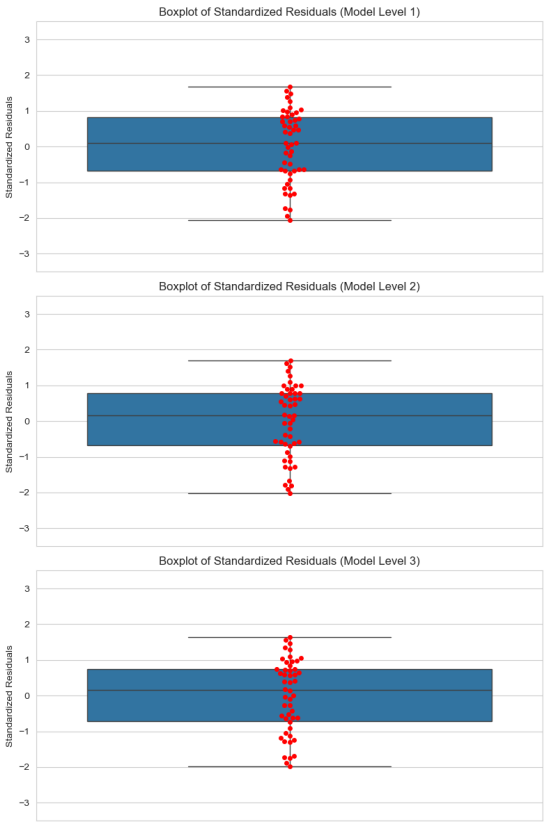
cues	VIF	Extreme Residuals	Cooks	Mean Residuals	SW
score	T	T	T	T	(F, 0.006)
score, replies	T	T	T	T	(F, 0.004)
score, replies, length	T	T	T	T	(F, 0.009)

Table 12: Statistical Tests for Hierarchy 1 - part 2

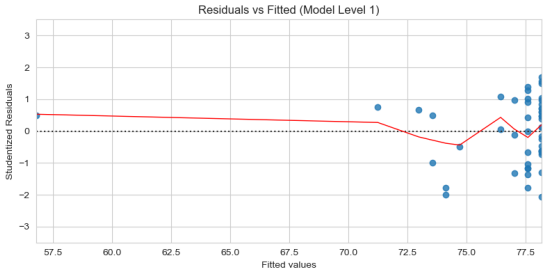
Plots of the graphs of different statistical values:

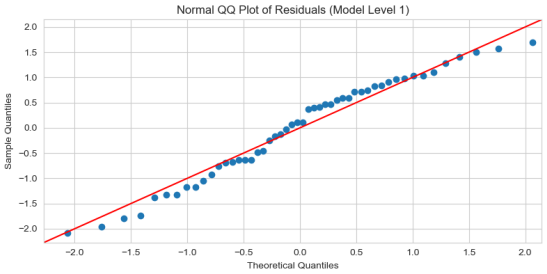


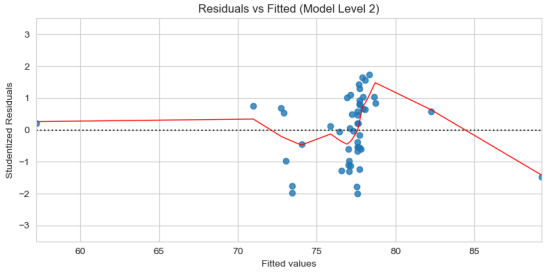
(a) Standardized Residuals Historgram

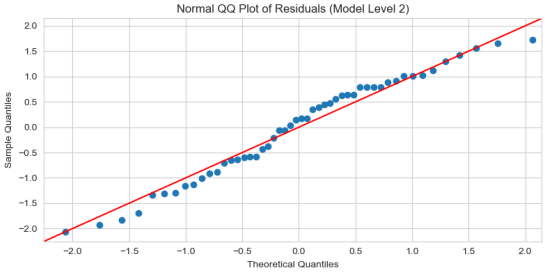


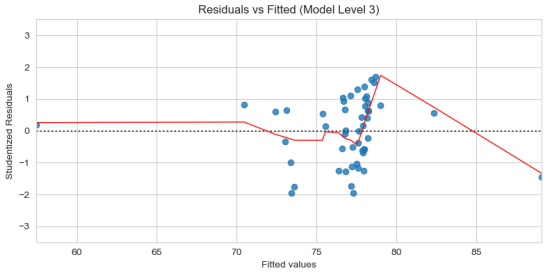
(b) Boxplot of Standardized Residuals

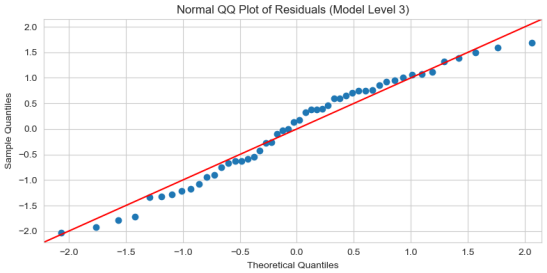












(a) Standardized Residuals plot

(b) QQ plot of Standardized Residuals

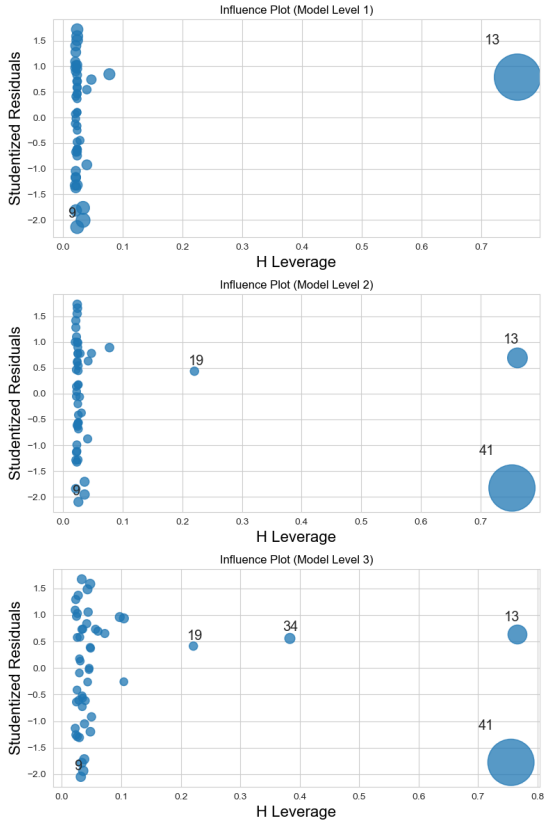
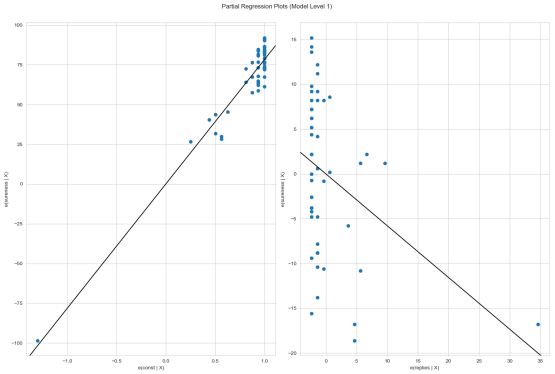
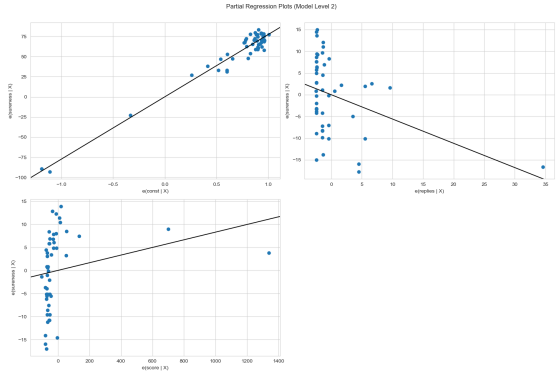


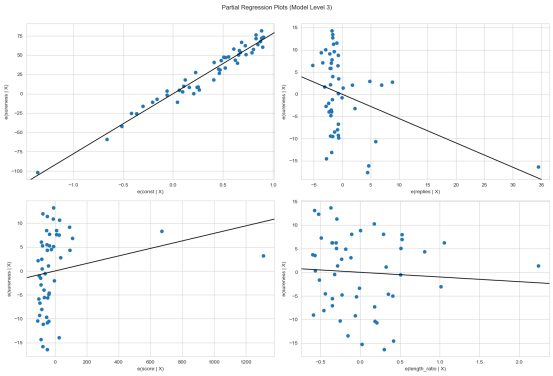
Figure 14: Plot of influence of Different Points on the model



(a) score



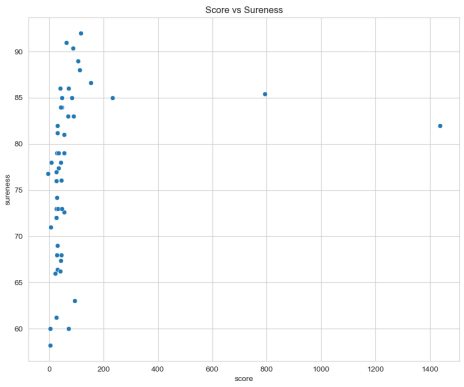
(b) score; replies



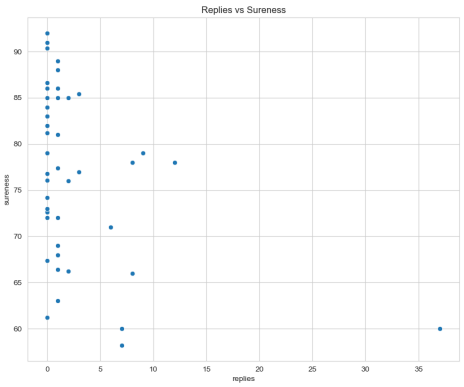
(c) score; replies; length

Figure 15: Plots of Standardized Residuals as a function of each individual cue + constant

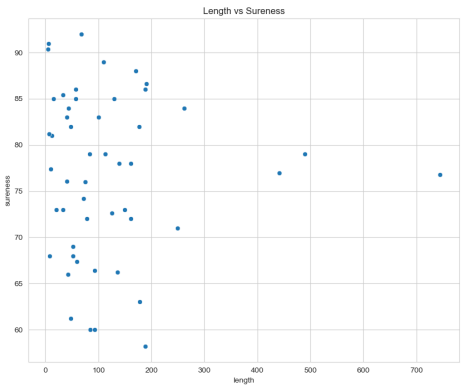
Plots of different cues and their effects on the final sureness:



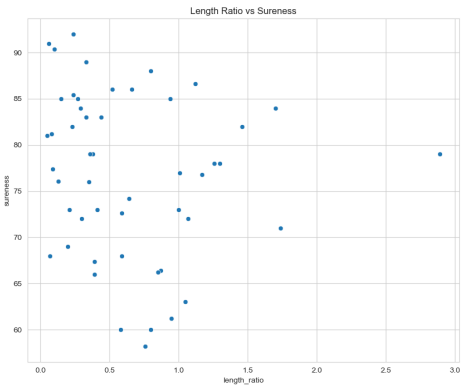
(a) Score - Sureness



(b) Replies - Sureness



(c) Length - Sureness



(d) Length Ratio - Sureness

Figure 16: Different Cues and their Effects on the final Sureness

5 Conclusions

5.1 Possible problems in data collection