ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Bachelor Thesis Econometrics (Financial)

# Using Utility Functions to predict human behaviour

Name Student: Marin Alkemade

Student ID: 507299

Supervisor: Aurelien Baillon

Second Assessor: Mikhail Zhelonkin

2 July 2022

**Abstract**

This paper aims to use the parametric approach for utility functions to predict human behaviour regarding gambles choices. The paper uses training sets to train these parameters to obtain the lowest Mean Squared Error. I will replicate part of the paper, (Peterson et al., 2021), and I will extend on this. I have investigated 9 different utility functions and I concluded that the 'Normalized Power' and the 'General Power' functions have the lowest Mean Squared Errors and thus perform the best in the individual model. In the extention part I have looked at adding a constant to the model and concluded that these utility functions were 60 percent of the time preferred over the constant. Lastly I made combinations of these utility functions within the model and concluded that having multiple functions within the model reduces the Mean Squared Errors. I also found that the proportion of the utility function used in the combination does seem to have an effect on the performance.

# Contents

# 1 Introduction

Trying to predict human behaviour is nowadays a common thing for big companies. Which customers to target and which advertisement to push is always the question and where there is no one answer to. Machine and Deep Learning can be then useful tools to predict human behaviour. For example in (Choi, Kim, & Oh, 2013) where they use deep learning to predict the behaviour of people to optimize smart home devices. However the functional form where Machine and Deep Learning is used on is often complicated and hard to understand. This is especially the case when non-parametric models are used. In this paper I try to predict human behaviour on gamble choices using simpler parametric models, where the functional form is less difficult to understand. These are models of utility functions. The goal is to see whether these simpler models are also effective in predicting human behaviour. If this is the case then these simpler models could also be used for a variety of bigger problems where now we would rely on complicated and difficult models to understand. In this paper I will replicate part of the paper *Using large-scale experiments and machine learning to discover theories of human decision-making* (Peterson et al., 2021). The part I will replicate is how the Mean Squared Error is computed using different testing samples as percentage of the data. I will use utility function models in the paper in an attempt to answers my first research question: *Can we accurately predict human behaviour using utility functions and which function is the most effective?*

Furthermore I will extend this part of the research and ask two more research questions. Firstly I will discuss whether there is a way to diversify the utility function from chance by using a chance factor. The chance factor is a constant of 0.5. I chose 0.5 as the constant because it assembles the assumption of the gamble choice being randomly chosen. Adding a parameter to the utility function and the chance factor will lead to an insight in what proportion of the time the function is being chosen over the constant of 0.5. The goal is to answer the question: *'Does the function better than complete chance and in what proportion? And does this lead to a lowering of the Mean Squared Error?'* We see that for the models this proportion of the function being chosen over complete chance is 60 percent. We also see that by adding the complete chance factor the Mean Squared Error also decreases.

Secondly I extend the model by using multiple utility functions in a single model and investigate whether this has an effect on the Mean Squared Error of the model. This model also includes the chance factor mentioned above. Here the research question would be: *'Does the use of multiple utility functions in the model improve the performance?'*. The answer to this question is yes, multiple utility functions do improve the performance. We will see that when using multiple function the Mean Squared Error goes down significantly, always outperforming both models individually.

To conclude, this paper consists of seven sections. In Section 2 I dive into the literature relating to our research. In Section 3 I give insight in the data and in Section 4 the methods and the models are explained. In Section 5 I discuss the results of these methods and data. Furthermore in Section 6 I will discuss shortcomings and further research ideas and lastly in Section 7 I conclude the paper with the overall conclusion of the results. References and additional information, such as an explanation of the code can be found at the end of the paper in Appendix A.

# 2 Literature

Expected Utility functions are very popular in the world of economics with a lot of uses such as for deciding an individual customer optimal need for goods and services. The first mentioning of such Expected Utility function was done by Bernoulli in 1738 regarding the St.Petersburg Paradox. This was a lottery where Bernoulli observed that people were hesitant to take the gamble on, even though the lottery had infinite expected value (Risk & BERNOULLI, 1954). In (Mongin, 1998) the expected utility theory states: 'that the decision maker chooses between risky or uncertain prospects by comparing the expected utility values, that is, the weighted sums obtained by adding the utility values of outcomes multiplied by their respective probabilities'.

## 2.1 The four axioms

For the utility function to work properly it has to hold for 4 axioms, the Von Neumann-Morgenstein axioms mentioned were discovered in 1944 and rewritten in (Von Neumann & Morgenstern, 2007). These are completeness, transitivity, independence of irrelevant alternatives and continuity. If the axioms hold then we can represent the preferences as an expected utility function.

Completeness means that for every choice between two options, either one is preferred to the other or they are indifferent. Secondly, transitivity means that if choice A is more preferable than choice B and choice B is more preferable than choice C than this would automatically mean that choice A is more preferable than choice C.

Furthermore, independence of irrelevant alternatives means that if choice A is more preferable than choice B, then a mixture of choice A and a choice C would be preferred to a similar mixture between choice B and choice C. However this axiom tends to be violated in practice. In a study done *Expected Utility Analysis without the Independence Axiom* (Machina, 1982) this is investigated. Machina found that the results of the expected utility function does not depend on this axiom but on a weaker assumption of smoothness. Lastly, continuity means that if choice A is preferred over B and choice B over C then there should be a combination of choice A and choice C where the decision maker would be indifferent between the combination and choice B.

## 2.2 Real Life uses

Parametric models are models where data is used to form a adequate model. Parametric models start with a form and use training data to train the models to use the right coefficients. This use of parametric models is very convenient, because these methods are easy to understand. These parametric models are for example used in modelling of survival data (Royston & Parmar, 2002). They found these parametric models to be better than the Cox proportional-hazards regression when analysing survival data. The advantage came especially when dealt with non-proportional hazards. Using these parametric models they gained better insights into the history of diseases and possible underlying causes of clinical events.

Utility functions themselves are also very useful for other problems. So is it widely used in economics in the supply and demand analysis in the form of indifference curves. These curves are made of combinations between different commodities where customers would gain the same utility (Allen, 1934). Moreover using the parametric approach for these utility functions could give benefits. These parametric utility functions are for example in use when decisions need to made under uncertainty. In the paper (Smith, 1972) they proposed to use parametric methods to obtain a

meaningful utility function that allows them to make decisions in the petroleum business. They do this by identifying key parameters within the business such as utility values and monetary values. The use of the parametric utility function offers solutions to problems where the normal expected value theory could not give a good decision. However parametric models are not always best to use. For example we also see that when it comes to option pricing that non-parametric models significantly outperform the parametric models. (Park, Kim, & Lee, 2014).

# 3 Data

In this section I will go over the data I used in this paper. The data is the same data used in (Peterson et al., 2021). The data is constructed in two parts, the gamble choices and the probabilities. I will firstly go over how the data is constructed regarding the gamble choices. Secondly I will discuss how the observed probabilities were obtained. In total the dataset has 14568 gamble choices and there are also 14568 observed probabilities, one for each gamble choice.

## 3.1 The gamble choices

The data consists of different gamble choices. In total the dataset has 14568 gamble choices where each choice consists of a gamble A and a gamble B. Gamble A and gamble B have probabilities $p_i$ and outcomes $x_i$ corresponding to the probabilities. Gamble A always has 2 probabilities and 2 outcomes. The amount of probabilities and outcomes for gamble B varies. But it has at least one outcome and one probability and at most it has 9 outcomes and 9 probabilities.

As an example you can see the first gamble choice in the dataset below;

$$\begin{array}{cc} \text{Gamble A} & \\ \begin{bmatrix} \text{Outcomes} & \text{Probabilities} \\ 26 & 0.95 \\ -1 & 0.05 \end{bmatrix} & \end{array} \quad \begin{array}{cc} \text{Gamble B} & \\ \begin{bmatrix} \text{Outcomes} & \text{Probabilities} \\ 21 & 0.95 \\ 23 & 0.05 \end{bmatrix} & \end{array}$$

What we can see is the outcomes do not have to be higher than 0. Also the number of probabilities and the probabilities themselves are in this example both the same but this does not have to be the case. The probabilities for each gamble however always should sum up to 1.

Furthermore the Expected Value of these gambles, given as $\sum_i x_i \cdot p_i$ does not have to be the same given a gamble choice. Here the EV for gamble A is 24.65 where the EV for gamble B is 21.1. The Expected Value is a simple mechanism to choose between gambles. In the Methodology and Results section I will dive deeper into these models evaluating different gambles.

### 3.2 The observed probabilities

Next I have the observed probabilities. These are the dependent variables in the optimisation. These probabilities were collected by the use of participants from Amazon Mechanical Turk. Each participant was paid 0.75 dollar and a bonus for the participation. This bonus was proportional to the reward from one gamble in their history at the end of experiment. For this reward each participant had to make a gamble choice for 20 choice problems. This was done in blocks of five trials per problem. Each trial the participant was asked to choose between the two gamble choices. So gamble A or gamble B. Based on this a variable was defined, *bRate*, which is the probability of a participant choosing gamble B over gamble A for a particular problem. I use the complement of this variable, so 1 - *bRate*, to get the probability of participants choosing gamble A over gamble B.

## 4 Methodology

In this section I will discuss how I used the data and which methods I used to come to the results. Firstly I will go over how and why I reduced the dataset. Secondly I will explain the replication part including the build up of the model, the functions being used and the optimisation technique. Lastly I will explain the extension part which goes over the diversification from chance and adding multiple functions to the model.

### 4.1 Reducing the data

The data set used in the paper (Peterson et al., 2021) was so large that the computing time for the optimisation was taking too long. Because of this I decided to use 10% of the data, this comes down to 1500 different gamble choices. The 1500 gamble choices were sampled randomly out of the bigger data set. This randomness ensures that the results are still adequate. All the analysis and

results you see below has been done on this dataset. So when I mention 'the dataset' in further sections, I will be discussing the 1500 sampled choices instead of the dataset consisting of the 14568 observations.

## 4.2 Replication

In this paper I will look at 9 out of the 12 utility models mentioned in (Peterson et al., 2021) and try to get the same Mean Squared Errors as they did. I choose 9 because the other 3 utility models did not find convergence or other errors occured when analyzed.

Firstly I will go over the different models that are to be analyzed. Secondly I explain how the model is set up and discuss the model into detail. Furthermore I will investigate how the parameters are estimated by minimizing the Training Mean Squared Error. The Training Mean Squared Error is calculated by Equation 3 on the training set. For the replication I used 10 percent, 25 percent, 50 percent, 75 percent and 100 percent as training set. This training set always consists of the first observations of the dataset, so for 25 percent I have the first 375 observations as training set. After estimating the parameters by minimizing Equation 3 for this training set, the parameters are used to calculate the Test Mean Squared Error. This Test Mean Squared Error calculation is done on the remaining data points that were not in the training set. When the training set is 100 percent then there are no observations left to compute the Test Mean Squared Error. To overcome this I take the Training Mean Squared Error as Test Mean Squared Error.

There is a complication that can occur while using this technique which is that the model can be overfitted. This means that the parameters I estimate from the training set give a low Mean Squared Error for only that set and not for the remaining test set. This is not desired. What you would want for a model to be good is to have the the Test Mean Squared Error slightly higher than the Training Mean Squared Error. I will compare both for all the utility functions to see whether the models are good. Of those good models the goal is to have the lowest Test Mean Squared Error. This model would be the best at predicting the chance of people choosing one gamble over the other.

### 4.2.1 The Utility Functions

In this section I discuss the 9 different utility functions. The functions range from simplistic linear function to more complex and normalized exponential and logarithmic functions. What you also see is that apart from the linear function all the functions are piecewise and symmetric around x = 0. Furthermore note that the lowest amount of parameters in a function is 1 and the highest amount is 5.

Linear: $u(x) = \lambda x$

Linear Loss averse: $u(x) = \begin{cases} x & \text{if } x \geq 0 \\ \lambda x & \text{if } x < 0 \end{cases}$

Power: $u(x) = \begin{cases} x^{\alpha} & \text{if } x \geq 0 \\ (-\lambda)(-x)^{-\beta} & \text{if } x < 0 \end{cases}$

Normalized Exponential: $u(x) = \begin{cases} \frac{1}{\alpha}(1 - e^{-\alpha x}) & \text{if } x \geq 0 \\ \frac{-\lambda}{\beta}(1 - e^{\beta x}) & \text{if } x < 0 \end{cases}$

Normalized Logarithmic: $u(x) = \begin{cases} \frac{1}{\alpha}\log(1 + \alpha x) & \text{if } x \geq 0 \\ \frac{-\lambda}{\beta}\log(1 - \beta x) & \text{if } x < 0 \end{cases}$

Normalized Power: $u(x) = \begin{cases} ((1 + \alpha)x)^{1+\alpha} & \text{if } x \geq 0 \\ -(-(1 + \beta)\frac{x}{\lambda})^{1+\beta} & \text{if } x < 0 \end{cases}$

Logarithmic: $u(x) = \begin{cases} \log(\alpha + x) & \text{if } x \geq 0 \\ -\lambda\log(-\beta x) & \text{if } x < 0 \end{cases}$

General Linear: $u(x) = \begin{cases} \alpha x & \text{if } x \geq 0 \\ \lambda\beta x & \text{if } x < 0 \end{cases}$

General Power: $u(x) = \begin{cases} \beta x^{\alpha} & \text{if } x \geq 0 \\ -\lambda(-\delta x)^{\gamma} & \text{if } x < 0 \end{cases}$

### 4.2.2 The build up of the model

In this section I will go through every component of the model. The model begins with the equation for the Expected Value. The equation calculates the sum of utilities weighted on the probabilities. $x_i$ is here the outcome of gamble i, with probability $p_i$. The function, $u(x_i)$, can be any of the 9 different utility functions mentioned before. The equation below represents the Expected Value for gamble A.

$$V(A) = \sum_i u(x_i)p_i \tag{1}$$

With the Expected Value I can calculate the probability of a participant choosing gamble A over gamble B. I denote the probability of choosing gamble A over gamble B with P(A) and can this be calculated by the equation below.

$$P(A) = \frac{e^{\eta V(A)}}{e^{\eta V(A)} + e^{\eta V(B)}} \tag{2}$$

Note that this P(A) is highly non-linear. Not only because of the exponential factor in it but also because it contains the Expected Value, V(A), which is also non-linear. The equation for P(A) is a very commonly used equation in Machine Learning called 'the softmax function'. This softmax function makes sure that every input of the Expected Value maps onto a number that is between 0 and 1 and that the sum of P(A) and P(B) is equal to 1.

The goal is to predict the decision of the gambler and so I want to minimize the distance between the P(A) that I get from this equation, denoted as $\hat{P}$(A), and the actual observed probability, P(A), I get from the data. This P(A) is the complement of the probability from the variable *bRate*. This can be done by using the Mean Squared Error as seen below. Here n are the individual observations and N is the total number of observations used in Equation 3

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_n [\hat{P}_n(A) - P_n(A)]^2 \tag{3}$$

### 4.2.3 Optimisation

For the optimisation of the parameters I use the Nelder-Mead Method. The Nelder-Mead method is commonly used for multidimensional unconstrained optimization (Singer & Nelder, 2009). An

advantage of using the Nelder-Mead method is it does not require a gradient to find the best solution, which is why it is perfect to use for non-smooth functions. I have chosen this method because of the complex nature of the model with the utility functions having a lot of parameters.

As said, the Nelder-Mead algorithm uses function output values to look to minimize the objective function. In this case the function to minimize is the Training Mean Squared Error mentioned in Equation 3. The Nelder-Mead algorithm is divided up into 6 steps and can be seen in Appendix B.

The optimisation will be done on part of the data set. I will investigate it for 10 percent, 25 percent, 50 percent, 75 percent and 100 percent of the data set. Every time I use the Nelder-Mead algorithm and use the estimated parameters to get the Test Mean Squared Error. In Table 1 we can see which parameters need to be estimated for each of the 9 utility functions.

Table 1: Parameters to be estimated for each utility function

| Model | Parameters |
|---|---|
| Linear | $\lambda, \eta$ |
| Linear Loss-Averse | $\lambda, \eta$ |
| Power | $\lambda, \alpha, \beta, \eta$ |
| Normalized Exponential | $\lambda, \alpha, \beta, \eta$ |
| Normalized Logarithmic | $\lambda, \alpha, \beta, \eta$ |
| Normalized Power | $\lambda, \alpha, \beta, \eta$ |
| Logarithmic | $\lambda, \alpha, \beta, \eta$ |
| General Linear | $\lambda, \alpha, \beta, \eta$ |
| General Power | $\lambda, \alpha, \beta, \gamma, \delta, \eta$ |

### 4.3 Extension

The extension consists of two parts. Firstly I will discuss the different utility functions using a complete chance factor in the model. Secondly I will estimate multiple utility functions at once instead of just one and see how they interact with each other. The goal for both of these methods is to reduce the Test Mean Squared Error.

### 4.3.1 Diversification from chance

The utility functions cannot capture all the essence of the observed probability and they do have an error factor in it. This error factor, or complete chance factor, can occur when participants have selected one of the two gambles randomly. Knowing this, I want to investigate to what extent does the utility function decide the outcome and to what extent is the outcome decided by this randomness.

I add the chance factor by using a constant of 0.5 as the random factor in the model as well as an extra parameter, $\theta$. The random factor is 0.5 because this has the interpretation that both gambles have equal chance of being chosen when selected randomly. The model for including this factor can be seen in Equation 4. The $\theta$ is the proportion of when the utility function is used and consequently then $1 - \theta$ is the proportion of when the random factor is used. As $\theta$ is a proportion, it should be between 0 and 1. Ideally I want $\theta$ to be as close to 1 as possible, indicating that the utility function is doing a good job in predicting the observed probability.

For this estimation I will also use the Nelder-Mead Algorithm. The training set samples will again be 10 percent, 25 percent, 50 percent, 75 percent and 100 percent.

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{n} [\theta \hat{P}_n(A) + (1 - \theta) \cdot 0.5 - P(A)]^2 \tag{4}$$

### 4.3.2 Multiple Utility Functions

One utility function cannot always capture the whole data set and so in this section I will look at what the effect would be if I were to use multiple utility functions in the model. The functions I will focus on will be the 4 best scoring functions from the diversification. This means that I will choose the functions that have the highest $\theta$ in the individual case. I do this because those functions have the highest proportion of explaining power and so would be best to use to increase the overall explaining power of the model. An example for a model with a combination looks like:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum_{n} [\theta_1 \hat{P}_{1,n}(A) + \theta_2 \hat{P}_{2,n}(A) + (1 - \theta_1 - \theta_2) * 0.5 - P_n(A)]^2 \tag{5}$$

Here $\hat{P}_{1,n}(A)$ is the estimated chance that someone chooses gamble A over gamble B for model 1. Same holds for $\hat{P}_{2,n}(A)$. The hope is that the sum of the $\theta$'s will be significantly higher than when using the utility functions individually and that this results in a lower Test Mean Squared Error.

Because the models have more parameters, the chance of overfitting increases. This could lead to a lower Training Mean Squared Error for the multiple utility function model but then a higher Test Mean Squared compared to the individual case. The amount of parameters will increase considerably, because now the model has #parameters of model 1 + #parameters of model 2 + 2 for $\theta_1$ and $\theta_2$. With the increase of parameters there arises a trade off between having more parameters and so having lower Training Mean Squared Error but also having longer computation times as well as the bigger chance of overfitting.

# 5 Results

In this section I will discuss the results. Firstly I will go through the replication part, seeing which estimates of parameters I get and investigate the characteristics of the Training and Test Mean Squared Errors. Furthermore whether different utility functions had significantly different results.

Moreover I investigate the results of the extension part where I look at the utility functions compared to chance and the multiple functions together to try to improve the Training and Test Mean Squared Errors obtained when using them individually.

## 5.1 Replication

### 5.1.1 Optimisation

When I performed the optimisation on Equation 3 using the Nelder-Mead algorithm, I got the following parameter estimates. You can see the values in Table 2 when using 100 percent as training set. They range between 0.02 and 0.035. The utility function with the lowest Training Mean Squared Error is 'General Power' with a Mean Squared Error of 0.0230. Also the 'Normalized Power' does well with a Mean Squared Error of 0.0234. What is surprising is that the normal 'Power' utility function does far worse having only a Mean Squared Error of 0.0316. Other func-

tions have higher Mean Squared Errors such as the Normalized Logarithmic with an Mean Squared Error of 0.0313. These estimations are in line with the findings in (Peterson et al., 2021) as seen in the figure in Appendix C where we can see the graph of the Mean Squared Errors.

Table 2: Estimations of parameters using 100 percent of the reduced data

| Utility function | Parameters to be estimated | Parameters | MSE |
|---|---|---|---|
| Linear | $\lambda, \eta$ | 0.098, 1.234 | 0.0284 |
| Linear Lose-Averse | $\lambda, \eta$ | 1.094, 0.119 | 0.0282 |
| Power | $\lambda, \alpha, \beta, \eta$ | 0.364, 0.789, 1.212, 0.860 | 0.0316 |
| Normalized Exponential | $\lambda, \alpha, \beta, \eta$ | 0.009, 0.022, 1.074, 0.148 | 0.0248 |
| Normalized Logarithmic | $\lambda, \alpha, \beta, \eta$ | 1.409, -0.003, 0.026, 0.985 | 0.0313 |
| Normalized Power | $\lambda, \alpha, \beta, \eta$ | -0.215, -0.182, 1.459, 0.333 | 0.0234 |
| Logarithmic | $\lambda, \alpha, \beta, \eta$ | 20.730, 0.000, 0.210, 3.116 | 0.0278 |
| General Linear | $\lambda, \alpha, \beta, \eta$ | 0.952, 0.770, 1.352, 0.125 | 0.0281 |
| General Power | $\lambda, \alpha, \beta, \gamma, \delta, \eta$ | 0.757, 0.835, 1.806, 0.246, 0.821, 0.351 | 0.0230 |

As mentioned in the Methodology section, these estimations are prone to overfitting, because I use all the data as training set to estimate the model. I investigated the Mean Squared Errors of the different models using different training sets to see whether there is an occurrence of overfitting. There is an occurrence of overfitting when the Test Mean Squared Error is significantly higher than the Training Mean Squared Error.

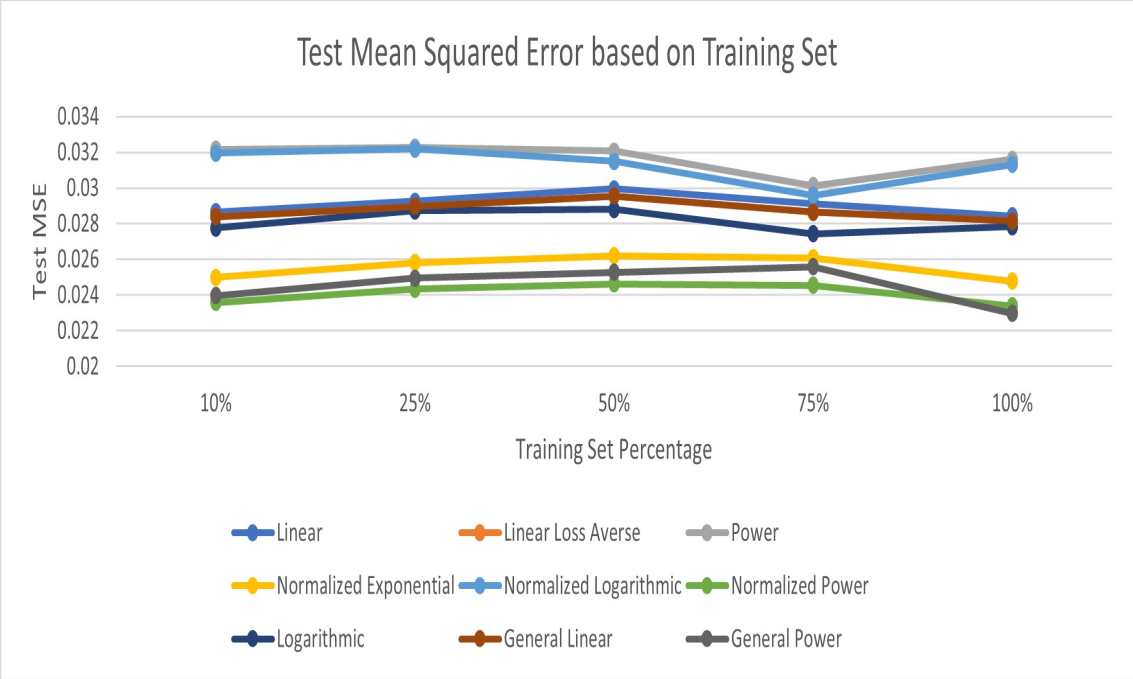
In Table 3 we see the values of the Training and Test Mean Squared Errors from the different utility functions using different training set percentages. When we look at the values that the Training and Test Mean Squared Errors produces we can take two things away from it. The first thing is that the values within an utility function do not massively vary when using different percentages as training sets. For example for the utility function 'Linear'.The Test Mean Squared Error changes for 10, 25, 50 and 75 percent from 0.0287 to 0.0293 to 0.03 to 0.0291. The steadiness of the Mean Squared Error does suggest that the utility functions can be well estimated and that the model does not need a lot of training observations to be effective. Besides the steadiness we can also see that in the models overfitting is not a problem. For almost all the models we can see in Table 3 that the Test Mean Squared Error is slightly higher than the Training Mean Squared Error. Indicating that the models are good.

Table 3: MSE Test versus MSE Sample

| | 10 percent | 25 percent | 50 percent | 75 percent |
|---|---|---|---|---|
| Linear | 0.0267 | 0.0261 | 0.0270 | 0.0282 |
| | 0.0287 | 0.0293 | 0.0300 | 0.0291 |
| Linear Loss Averse | 0.0263 | 0.0260 | 0.0268 | 0.0280 |
| | 0.0284 | 0.0290 | 0.0300 | 0.0286 |
| Power | 0.0299 | 0.0297 | 0.0312 | 0.0321 |
| | 0.0321 | 0.0323 | 0.0321 | 0.0301 |
| Normalized Exponential | 0.0237 | 0.0225 | 0.0235 | 0.0244 |
| | 0.0250 | 0.0258 | 0.0262 | 0.0261 |
| Normalized Logarithmic | 0.0294 | 0.0293 | 0.0312 | 0.0319 |
| | 0.0320 | 0.0322 | 0.0315 | 0.0296 |
| Normalized Power | 0.0222 | 0.0210 | 0.0223 | 0.0231 |
| | 0.0236 | 0.0243 | 0.0246 | 0.0245 |
| Logarithmic | 0.0281 | 0.0249 | 0.0270 | 0.0276 |
| | 0.0278 | 0.0287 | 0.0288 | 0.0274 |
| General Linear | 0.0263 | 0.0260 | 0.0268 | 0.0280 |
| | 0.0284 | 0.0290 | 0.0296 | 0.0286 |
| General Power | 0.0210 | 0.0194 | 0.0212 | 0.0222 |
| | 0.0240 | 0.0250 | 0.0253 | 0.0256 |

So I do not have a reason to say that the models are overfitted. In Figure 2 I have plotted the Test Mean Squared Error for the values from Table 3. In the figure we can see that the 'Normalized Power' and the 'General Power' are the best in terms of Mean Squared Error. We also see, as I mentioned before, the difference from these two functions to the standard 'Power' function. This model, together with 'Normalized Logarithmic', performs the worst out of all these 9 functions. Also note here that that the Mean Squared Errors do not chance significantly when using different percentages as Training sets.

Figure 1: The Test Mean Squared Error compared to Training set used




## 5.2 Extension

In this section I will look at the results of the extension parts. I will discuss what the results are when a chance factor is introduced and what effect it has if multiple functions are used simultaneously in the model.

### 5.2.1 Diversification from chance

In Table 4 we can see the results of the model from Equation 4 where the chance factor is implemented. Especially when we look at the $\theta$ we can see that 'Normalized Power' and 'General Power' are the two best utility function in terms of the proportion of being chosen over complete chance. These utility functions are also the ones with the lowest in terms of Test Mean Squared Error. Also for the third best utility function, in terms of $\theta$, 'Normalized Exponential', it holds that it is third best in Test Mean Squared Error. I cannot do an adequate regression analysis as I only have 9 observations but I can hypothesize that a higher $\theta$ causes a better performance of the Test Mean Squared Error.

Furthermore what we see is that the difference between the Training Mean Squared Error and the Test Mean Squared Error is slightly lower for most utility functions when we have the chance factor included in the model. For example, when we look at Table 4, the difference at 50 percent training set for 'Linear Loss-Averse' with the chance factor is 0.025 and for the model without 0.032. This is odd because you would expect that with adding more parameters such as $\theta$ that overfitting would be more of a problem but it looks like the opposite happens when 8 of the 9 have this outcome. I did not test for the significance of the differences so this could be due to chance but it is interesting to mention.

Table 4: Test MSE with chance factor (training set at 50 percent)

| Utility Function | $\theta$ | Other Parameters Estimation | MSE Training | MSE Test |
|---|---|---|---|---|
| Linear | 0.582 | 1.934, 0.136 | 0.0267 | 0.0295 |
| Linear Loss-Averse | 0.572 | 1.079, 0.266 | 0.0265 | 0.0290 |
| Power | 0.539 | 0.409, 0.720, 0.928, 1.899 | 0.0305 | 0.0311 |
| Normalized Exponential | 0.584 | 0.010, 0.032, 1.193, 0.326 | 0.0230 | 0.0254 |
| Normalized Logarithmic | 0.328 | 3.009, 0.092, 0.038, 39.927, 0.328 | 0.0281 | 0.0274 |
| Normalized Power | 0.606 | -0.197, -0.259, 0.863, 0.628, 0.605 | 0.0219 | 0.0239 |
| Logarithmic | 0.507 | 19.647, 0.000, 0.226, 9.107 | 0.0260 | 0.0271 |
| General Linear | 0.572 | 0.128, 0.324, 0.425, 2.088 | 0.0265 | 0.0290 |
| General Power | 0.627 | 0.748, 0.766, 2.173, 0.197, 0.766, 0.759 | 0.0209 | 0.0247 |

#### 5.2.2 Multiple utility function

Now I will look at the what happens when I use multiple utility functions in the model. For this I used Equation 5. I will first discuss the behaviour of the $\theta$'s and to compare them to the individual function models. Secondly I will discuss what happens with the Training and Test Mean Squared Error. Lastly I would like to point out that the combination Linear/Normalized Power did not find convergence so this combination will not be further analyzed in this section.

When we look at Table 5 we can see that the sum of $\theta$'s did increase for 4 out of the 5 combinations. The only combination that did not have an increase compared to the $\theta$'s for the individual model is 'Linear / G.Power'. This means that for the rest of the models the overall proportion of the functions being favoured over the complete chance is greater. This is positive because it shows that with more choice between different functions within the model, the proportion of people where both

of the models are ineffective gets smaller. Another thing we can see is that for the two combined proportions that are really high, 'Norm Exp. / G Power' and 'Norm Power. / G Power', one function is really dominate over the other. In both cases the G. Power function has a $\theta$ far greater than the other function in the model. This means that the G.Power function is better than the other function at predicting all sorts of the gamble choices. This is less true when we look at the combination of 'Linear/G.Power' where the $\theta$'s are more evenly split and G.Power is even the lower of the two. This distribution of $\theta$'s being more even could suggest that there is a specific group of gamble choices where the Linear is better at evaluating and there is a specific group of gamble choices where the General Power function is better. I did not analyze whether this is the case but an example of such a group could be; all gamble choices where both gambles only have 2 outcomes the Linear function performs better and all gamble choice where there are more than 2 outcomes the General Power does.

Table 5: MSE of the model for different training sets

| | $\theta_1$ | $\theta_2$ | Combined | G Power | Norm. Exp | Norm. Power | Linear |
|---|---|---|---|---|---|---|---|
| Norm. Exp. / G.Power | 0.280 | 0.560 | 0.840 | 0.627 | 0.584 | | |
| Norm.Power / G. Power | 0.153 | 0.596 | 0.749 | 0.627 | | 0.606 | |
| Linear / G.Power | 0.394 | 0.228 | 0.622 | 0.627 | | | 0.582 |
| Norm. Power / Norm. Exp | 0.149 | 0.485 | 0.634 | | 0.584 | 0.606 | |
| Linear / Norm. Exp | 0.424 | 0.178 | 0.602 | | 0.584 | | 0.582 |

Table 6 consists of the Training and Test Mean Squared Errors of the combinations. It also includes the Mean Squared Errors of the individual models. Here 'MSE 1' is the first function in the first column so in the first row this would be 'Norm Exp' and the 'MSE 2' is second function and so in the first row this would be 'G.Power'.

When we not only look at the $\theta$'s but also at the Mean Squared Errors we can see something different. We see that the combination of 'Norm.Power/General Power' does best in terms of Training Mean Squared Error but does not so well compared to the others when looking at the Test Mean Squared Error.

Table 6: Training and Test MSE of the model combined and individually at 50 percent training set

| | Training | | | Test | | |
|---|---|---|---|---|---|---|
| | Comb. MSE | MSE 1 | MSE 2 | Comb. MSE | MSE 1 | MSE 2 |
| Norm. Exp. / G.Power | 0.0207 | 0.0230 | 0.0209 | 0.0244 | 0.0254 | 0.0247 |
| Norm.Power / G. Power | 0.0188 | 0.0219 | 0.0209 | 0.0230 | 0.0239 | 0.0247 |
| Linear / G.Power | 0.0199 | 0.0267 | 0.0209 | 0.0220 | 0.0295 | 0.0247 |
| Norm. Power / Norm. Exp | 0.0208 | 0.0219 | 0.0230 | 0.0221 | 0.0239 | 0.0254 |
| Linear / Norm. Exp | 0.0209 | 0.0267 | 0.0230 | 0.0220 | 0.0295 | 0.0254 |

Interesting to note is that the combinations who have the lowest combination proportion of explanatory power, $\theta_1 + \theta_2$ in Table 5, have the lowest Test Mean Squared Errors. What we also see is that for the combination where one utility function was very dominant (one $\theta$ being very high compared to the other) the Test Mean Squared Error do not significantly improve. A good example is the combination of 'Norm.Exp./General Power' where General Power dominates but this results in that the Test Mean Squared Errors of the combination model are really close to the Test Mean Squared Errors of the individual model with only General Power in it. This holds also for the combination 'Norm.Power/G.Power'. This could be due to the dominance of one function the gain of having the other function is limited, because not a lot of observations are to be analyzed by this less effective function.

To conclude, I have seen in the individual case that when there is a higher proportion of explanatory power, the $\theta$, then the Mean Squared Errors are lower. This is however not the case for the multiple utility function model where the distribution of the $\theta$'s more impact has on the Mean Squared Errors. For the Individual case the General Power function is the best utility function to use to predict gamble choices and for the multiple utility function case I would choose the Linear/General Power combination as it has the lowest Test Mean Squared Error and second lowest Training Mean Squared Error.

# 6 Discussion

In this section I discuss a shortcoming of this research and suggestions for further research. The shortcoming is that I used only a part of the data due to computation times. The results were however adequate enough for the replication part, because they were in the same interval as they

were in the original paper. The figure of this is shown in Appendix C. The long computation times did have another big drawback, namely that now the big data set is not used for the extension parts. Especially for the use of multiple models it could be good to try to use the whole data set of more than 14000 gamble choices and see whether this would lead to the same results.

Another topic for further research would be to look into not only combinations of functions but also in models where there are three or more functions within a single model. Here a trade off would occur between more accurate results because of more parameters and more computation times.

# 7 Conclusion

To conclude this paper, I have investigate the Mean Squared Errors of 9 different utility functions based on gamble choices. For the research question: *Can we accurately predict human behaviour using utility functions and which function is the most effective?* I concluded that the we can accurately predict human behaviour and that the 'General Power' function had the lowest Mean Squared Errors and so was the best in the individual utility function model. Furthermore no model suffered from overfitting.

Secondly for the two research questions regarding the extension part, I conclude that when using a chance factor in the model the proportion of the utility function being used over the chance factor was around the 60 percent. Also it seemed that the higher the proportion the lower the Test Mean Squared Errors were. Here the 'General Power' was again the best in predicting the probabilities. Lastly I concluded that using combination of different utility functions does improve the performance of the prediction and reduces the Test Mean Squared Error. The distribution of the functions used in the model does however matter in terms of lowering the Test Mean Squared Error. I concluded that a more evenly distribution of methods does give better results than when one function dominates over the other.

# References


Allen, R. G. D. (1934). The nature of indifference curves. *The Review of Economic Studies*, *1*(2), 110–121.

Choi, S., Kim, E., & Oh, S. (2013). Human behavior prediction for smart homes using deep learning. In *2013 ieee ro-man* (pp. 173–179).

Machina, M. J. (1982). "expected utility" analysis without the independence axiom. *Econometrica*, *50*(2), 277–323.

Mongin, P. (1998). Expected utility theory.

Park, H., Kim, N., & Lee, J. (2014). Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over kospi 200 index options. *Expert Systems with Applications*, *41*(11), 5227–5237.

Peterson, J. C., Bourgin, D. D., Agrawal, M., Reichman, D., & Griffiths, T. L. (2021). Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, *372*(6547), 1209–1214.

Risk, O., & BERNOULLI, D. (1954). Exposition of a new theory on the measurement. *Econometrica*, *22*(1), 23–36.

Royston, P., & Parmar, M. K. (2002). Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in medicine*, *21*(15), 2175–2197.

Singer, S., & Nelder, J. (2009). Nelder-mead algorithm. *Scholarpedia*, *4*(7), 2928.

Smith, M. B. (1972). *Parametric Utility Functions for Decisions Under Uncertainty.*

Von Neumann, J., & Morgenstern, O. (2007). Theory of games and economic behavior. In *Theory of games and economic behavior.* Princeton university press.

# Appendix A

The code is build up into 16 different files. One file is for the data, then each utility function has its own file and each combination has it's own file. For the individual models the files contain the Training Mean Squared Error and Test Mean Squared Error calculation. This is for the one with and without the chance factor in it. The optimisation is done by the *optimizer.function* with *param* as starting point and *nmb* as the first observation that is not in the training set. The starting point for the parameters is 1 for each parameter except for the proportion (when using the chance factor model), this is set at 0.5. The only exception to this is the model Normalized Power where all this starting point was not viable and so all the parameters were 0.5.

The first function of these 9 files formulates the normal model and the *chance.function* formulates the model with the parameter. When using the *optimizer.function* I changed the first function with the *chance.function* when I want to optimize the other function. The *MSE.function* calculates the Test Mean Squared Error of the parameters given by the optimisation process. Here the row were *func* is calculated should be the one above for the normal model and the one below for the *chance.function*.

Lastly the other 6 files contain the combination model. For the combination models the method is the same. The first function formulates the model and the *optimizer.function* optimizes this model and gives the Training Mean Squared Error. Using the *MSE.function* you get the Test Mean Squared Error. Here again the starting parameters are all 1 except for the $\theta$'s where it is 0.5. And again when Normalized Power is in the combination all starting parameters are equal to 0.5.

# Appendix B

**Nelder-Mead Algorithm**

**1.** Order the current test points: $x_1, x_2, ..., x_{n+1}$ where $f(x_1) \leq f(x_2) \leq ... \leq f(x_{n+1})$.

**2.** Calculate the centroid excluding $x_{n+1}$ with $\sum_i^n x_i$

**3.** Calculate reflected point: $x_r = x_0 + \alpha(x_0 - x_{n+1})$

If $f(x_1) \leq f(x_r) < f(x_{n+1})$

then replace $f(x_r)$ with $f(x_{n+1})$ and go back to step 1 otherwise step 4.

**4.** If $f(x_1) > f(x_r)$ then compute the expanded point $x_e = x_0 + \gamma(x_r - x_0)$ with $\gamma > 1$

If $f(x_e) < f(x_r)$ then obtain new point by replacing $f(x_{n+1})$ with $f(x_e)$ and go to step 1.

else replace $f(x_r)$ with $f(x_{n+1})$ and go back to step 1

**5.** If $f(x_r) < f(x_{n+1})$ then compute the contracted point on the outside, $x_c = x_0 + \rho(x_r - x_0)$ with $0 < \rho \leq 0.5$

If $f(x_c) < f(x_r)$ then replace the worst point with the contracted point and go to step 1, else go to step 6.

If $f(x_r) < f(x_{n+1})$ then compute the contracted point on the inside, $x_c = x_0 + \rho(x_{n+1} - x_0)$ with $0 < \rho \leq 0.5$

If $f(x_c) < f(x_{n+1})$ then replace the worst point with the contracted point and go to step 1, else go to step 6.

**6.** Replace all points except $x_1$ with $x_i = x_1 + \sigma(x_i - x_1)$ and go to step 1.

# Appendix C

Figure 2: The Test Mean Squared Error compared to Training set used