**(7089CEM)**

Coursework

Introduction to Statistical Methods for Data Science

Module Leader: **Prof. Fei He**

Student Name: **Guddu Kumar Shah**

SID: **11402949**

**Modelling brain signals using nonlinear regression**

I can confirm that all work submitted is my own: Yes

1. **Introduction**

Statistics is an integral part in the data science and is one amongst the most prominent field which provides tools and methods to find hidden patterns, provide much deeper insights into data and quantifying uncertainty. Statistical analysis is a technique of collecting data from different sources followed by organizing, interpreting, analyzing and then deriving insights that helps in growing of the businesses. The main concept behind this is to make complex and unstructured data easy the read and comprehend. Nowadays, almost every industry like medical, marketing, engineering, autonomous vehicles, smartphones, stocks exchange are heavily data-driven and is creating humongous amount of data every single day. Therefore, it is very important to analyze the data sources, cleaning and then processing the cleaned data and understand the main reason behind the data generation, find insights and make changes for increasing organizations business. For performing such quantitative analysis, statistics comes out to become a powerful tool.

There statistical analysis of the data for different sources and be carried out in the following stages.

1. Defining the problem statement
2. Collecting data
3. Organizing data
4. Analysis of data
5. Interpretation of the analysis
6. Results-oriented data

The above steps will be followed during this study which likely resembles methods used by different scientists in their past works. The significance of statistics in data science lies in finding out the most important features for model building, best practices to conduct the experiment, designing best strategy on the basis of experiment’s output, best performance metrics and differentiating valid and noisy data.

1. **Nonlinear Regression**

Nonlinear regression is basically a type of regression analysis in which data is defined into mathematical function after fitting it to the model. It differs from linear regression in which two variables X and Y makes a straight line *y = mx+b* while nonlinear regression makes a curved relationship between the variables (Kenton, 2022). The mathematical expression for a nonlinear regression model can be written as

*y = (X, β) + ε*

where:

* X is a vector of P parameters
* β is a vector of k variables
* F (-) is the known regression function
* *ε* is the error sum

The nonlinear model can also be represented as :

Where:

* Yi is the responsive variable
* h is the function
* x is the input
* θ is the parameter to be estimated

In the above expression Yi can contain the combination of linear as well as nonlinear parameters and each of these parameters can be tested to determine the model to be linear or nonlinear. The function h is taken into consideration as it cannot be written as linear. The nonlinear parameters in the model is just opposite to the independent variables. Because of the nonlinear parameters there exists immense number of possibilities for evaluating the deterministic part of the model. This flexibility provides a very strong base to perform statistical inferences. The main aim of this model is to reduce the sum of the square as least as possible using iterative numeric procedures. The sum of squares is basically used to determine the fitness of a regression model by calculating the difference between mean and all data points. Then, each differences is squared prior to summing all the squared data. This sum finally determines how a model best fits the data, the smaller the better the model fits the data. The goodness of the fit using the computed least squares determines how well the curve fits. In other words, how well the curve fits the data is obtained from the magnitude of the differences between the curve and the data points.

1. **Aim and Dataset Description**

The main aim of this project is to select the best regression model from a candidate set of nonlinear regression models that perfectly describes the brain response to the sound signals. The data were collected through a neuromarketing experiment where participant was given a task to listen to advertisement and investigate how their brain responded using MEG (Magnetoencephalography). MEG is considered as a non-invasive discipline for investigation of brain activity by measuring the magnetic field generated by electrical activity of neurons. It tracks the ongoing brain activity on a millisecond-by-millisecond basis and captures where in the brain activity is produced (Singh, 2014). MEG is specifically recorded from the amygdala, a region in the brain for emotion processing. In this experiment, in the first 10 seconds a participant listens to an advertisement in neutral voice and in the next 10 seconds another advertisement in emotional voice. The regression model which perfectly measures the audiotory-brain interraction and which shows significant effect of emotional narration will be identified in this study. This research work will provide researchers a hypothesis whether the emotional narration enhance the brain response.

Three files (X.csv, y.csv and time.csv) were used for this study. The MEG time-series data and the sound signal are presented in two files X.csv and y.csv. The X.csv file contains two features x1 and x2. x1 is the input sound signal and x2 is the category of audio played during advertisements (x2 = 0 for neutral audio and x2 = 1 for emotional audio). During the experiment x2 was noted 0 when the neutral audio was played whilst x2 was noted 1 when the emotional audio was played. The y.csv file has one feature y which is the output MEG signal for different input sounds. The time.csv file has the data of sampling time for both input sound signal from X.csv and output MEG signal from y.csv. Furthermore, the output sound signal was added with (Gaussian with zero-mean) noise due to distortions generated during recording.

1. **Experimental Setup**

The project was demonstrated using Python programming language which is rich in machine learning libraries. All the code is written in juputer Notebook (interactive computer environment) that supports IPython command shell for Python and many more languages like Julia, R, Ruby, Haskell, Scala, node.js.The project runs Python 3.11.0 on Jupyter Notebook.

Prior to the experimental operation, several machine learning libraries as well as predefined functions are imported on the jupyter notebook. The main work of this project project is to select the best regression model from a candidate set of nonlinear regression models that perfectly describes the brain response to the sound signals.

After that pandas library is used to load the the three files X.csv, y.csv and time.csv which are in the excel format. These 3 excel files are combined to form a dataframe and is assigned to *df* variable of pandas dataframe. To confirm that the dataset has been loaded properly the dataframe variable *df is* printed data with all the attributes represented in *Figure 1*.

Table

Description automatically generated

Figure 1: Complete dataframe

(Source by Author)

**Task 1: Preliminary Data Analysis**

**1.1: Time series plots (of input audio and output MEG signal)**

After loading the dataset in pandas dataframe, the preliminary data analysis is performed. The task here is to plot the time series plots for input signal and output MEG signal. The time series plot for input audio signal is plotted concerning x1 column and time column from the dataframe as shown in *Figure 2.* The total number of samples is 200. It can be clearly seen that the input audio signal lies in the scale of -3 to 3 and time varying from 0 to 20 with 0.1 intervals.

Chart

Description automatically generated

Figure : Time series plot for audio x1

(Source by Author)

The time series plot for input audio signal is plotted concerning y column and time column from the dataframe as shown in *Figure 3.* The total number of samples is 200. It can be clearly seen that the output audio signal lies in the scale of 0 to 70 and time varying from 0 to 20 with 0.1 intervals.

Chart

Description automatically generated

Figure : Time series plot for out MEG signal

(Source by Author)

**1.2: Distribution for each (input & output) signal**

The distribution of input audio signal (column x1) is plotted concerning the input audio data. The plot is shown in *Figure 4*. The input signal lies on the scale of -3 to 3. Distribution plot is used to measure the counts of each input audio data. According to the figure, it can be clearly observed that the input audio data which varied from 0.5 to 1 occurred the most compared to other occurrences.

Chart, histogram

Description automatically generated

Figure : Distribution of input audio signal

(Source by Author)

The distribution of input audio signal (column x1) is plotted concerning the input audio data. The plot is shown in *Figure 5*. The input signal lies on the scale of 0 to 70. Distribution plot is used to measure the counts of each output MEG signal. According to the figure, it can be clearly observed that the output MRG data which varied from 10 to 20 occurred the most compared to other occurrences.

Chart, histogram

Description automatically generated

Figure : Distribution plot of output MEG signal

(Source by Author)

**1.3: Correlation and scatter plots (between the audio input and output brain signal) to examine their dependencies**

The correlation plot between the audio input signal and output brain signal is depicted is fig for examining their dependencies. The correlation matrix shows that the input signal x1 is highly correlated to the output MEG signal i.e., 0.77. Additionally, input audio signal x1 shows high positive correlation with the x2 feature and time feature which are 0.68 and 0.87.

The scatter plot between audio input and output MEG signal is depicted in figure for examining dependencies. From the figure, it can be clearly seen that if the input audio signal is between -3 to 1 the output MEG signal will be in the range of 0 to 30 and upon further increasing the input audio signal the output MEG signal increases respectively.

|  |  |
| --- | --- |
| **Chart, treemap chart  Description automatically generated**  Figure : Correlation plot between input and output brain signal  (Source by Author) | **Chart, scatter chart  Description automatically generated**  Figure : Scatter plot between audio input and output brain signal  (Source by Author) |

**1.4: boxplots of output brain signals to examine effect of sound categories**

The boxplot for output brain signal for examining the effects of sound categories is plotted in figure. By analyzing the box plot we can say that the median lies at 15, inter quartile ranges between 10 to 25 where lower quartile lies at 10 and upper quartile at 25. The min value is 1 whilst the max is 42.

Chart, box and whisker chart

Description automatically generated

Figure : Box plot for output MEG signal

(Source by Author)

**1.5: You can perform the above preliminary data analysis for each type of input sound signal separately (i.e. when 𝑥2 = 0, and 𝑥2 = 1)**

* Preliminary data analysis for input audio signal when x2 = 0 (neutral audio)

For this analysis the dataset is divided into two and the column x1 which is a neutral audio signal or x2 = 0 is used in this section of the work. The time series plots for both the input audio signal and output MEG signal is depicted in the *Figure 9 and Figure 10*. From figure is the input audio signal for neutral audio is distributed on the scale of -3 to 3 and output MEG signal from 0 to 70. The time is varies from 1 to 10 and the total number of samples taken is 100.

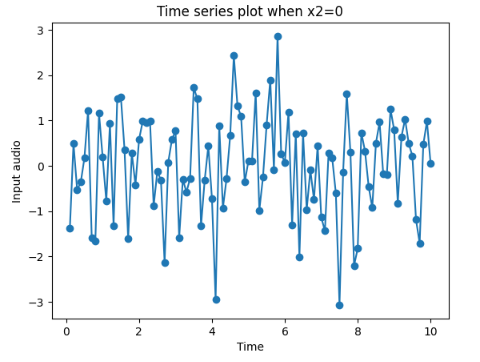


Figure : Time series plot for input audio when x2 = 0

(Source by Author)

Chart, scatter chart

Description automatically generated

Figure : Time series plot for output MEG signal when x2 = 0

(Source by Author)

From distribution plots of input audio signal and output MEG signal when x2 = 0 or when the audio is neutral is shown in *Figure 11* and *Figure 12*. For input audio, it can be said that input between 1 to 1.5 has the greatest number of occurrences.

Chart, histogram

Description automatically generated

Figure : Distribution plot for input when x2 = 0

(Source by Author)

For output signal, the distributed plot shows output has the highest occurrences between 10 to 15.

Chart, histogram

Description automatically generated

Figure : Distribution plot for output signal when x2 = 0

(Source by Author)

The scatter plot for input audio and output MEG signal when audio is neutral is shown in *Figure 13*. From the plot, if the input audio signal is between -3 to 1 the output MEG signal will be in the range of 0 to 30 and upon further increasing the input audio signal the output MEG signal increases respectively.

Chart, scatter chart

Description automatically generated

Figure : Scatter plot, x2 = 0

(Source by Author)

The correlation matrix in *Figure 14* shows that the input audio signal when it is neutral shows positive correlation with the output MEG signal i.e., 0.7.

Chart, waterfall chart

Description automatically generated

Figure : Correlation matrix, x2 = 0

(Source by Author)

The data analysis for neutral audio signal is shown in *Figure 15*. The median lies at 15, inter quartile ranges between 1 to 22, lower and upper quartile at 10 and 22.

Chart, box and whisker chart

Description automatically generated

Figure : Box plot for output MEG signal, x2 = 0

(Source by Author)

* Preliminary data analysis for input audio signal when x2 = 1(emotional audio)

For this analysis, the column x1 which is a emotional audio signal or x2 = 1 is used in this section of the work. The time series plots for both the input audio signal and output MEG signal is depicted in the *Figure 16* and *Figure 17*. From figure is clear that the input audio signal for emotional audio is distributed on the scale of -2.5 to 2.5 and output MEG signal from 0 to 60. The time is varies from 10 to 20 and the remaining total number of samples taken is 100.

Chart

Description automatically generated

Figure : Time series plot for input audio, x2 = 1

(Source by Author)

Chart

Description automatically generated

Figure : Time series plot for output MEG signal, x2 = 1

(Source by Author)

From distribution plots of input audio signal and output MEG signal when x2 = 1 or when the audio is emotional is shown in *Figure 18* and *Figure 19.* For input audio, it can be said that input between 0 to 0.5 has the greatest number of occurrences.

Chart, histogram

Description automatically generated

Figure : Distribution plot for input signal, x2 = 1

(Source by Author)

For output signal, the distributed plot shows output has the highest occurrences between 5 to 13.

Chart, histogram

Description automatically generated

Figure : Distribution plot for output MEG signal, x2 = 1

(Source by Author)

The scatter plot for input audio and output MEG signal when audio is emotional is shown in *Figure 20*. Here if the input audio signal is between -2 to 1 the output MEG signal will be in the range of 0 to 30 and upon further increasing the input audio signal the output MEG signal increases in correspondence.

**Chart, scatter chart

Description automatically generated**

Figure : Scatter plot for input and output MEG signal, x2 = 1

(Source by Author)

The correlation matrix in *Figure 21* shows that the input audio signal when it is emotional shows positive correlation with the output MEG signal i.e., 0.85.

**Chart, waterfall chart

Description automatically generated**

Figure : Correlation matrix, x2 = 1

(Source by Author)

The data analysis for neutral audio signal is shown in *Figure 22*. The median lies at 15, inter quartile ranges between 12 to 22, lower and upper quartile at 12 and 22.

**Chart, box and whisker chart

Description automatically generated**

Figure : Boxplot for output MEG signal, x2 = 1

(Source by Author)

**Task 2: Regression – modelling the brain response (MEG) to a sound signal**

The main objective of this task is to find the best model from the candidate set of models that describes the relationship between the input audio signal and output brain signal. This relationship is described by a polynomial regression model. The mathematical model also evokes how the relationship between the input and the output MEG signals changes with the change in the categories of input audio signal which are neutral and emotional. There are in total 5 polynomial regression expression each representing a nonlinear regression model and one out of them truly describes the model. The best model out of remaining models is needed to be identified for modelling the brain response to a sound signal.

The set of candidate models are represented in a mathematical expression below.

Model 1:

Model 2:

Model 3:

Model 4:

Model 5:

**Task 2.1**

In this task, the model parameters for every candidate model is estimated using least Squares (). The parameter estimation will be done using the input sound and out MEG signals and all the data will be used for training purposes.

The model parameter can be expressed as

Least Squares can be expressed as

The estimated model parameters are for all candidate models is set in *Table 1* below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| Model 1 | 5.17127 | -0.5373 | 2.20201 |  |  | 16.9357 |
| Model 2 | 8.62910 | 2.8139 |  |  |  | 16.6355 |
| Model 3 | 8.07945 | 0.31445 | 0.55920 | 3.96814 |  | 14.1317 |
| Model 4 | 9.3539 | 4.6469 | -0.57792 | 0.07479 | 4.34321 | 10.85328 |
| Model 5 | 8.55216 | 6.24691 | -0.28296 | 4.15988 |  | 10.17369 |

Table : Model parameters for 5 candidate models

A regression plot between input audio signal and output MEG signal for all the five candidate models are plotted below.



A regression plot between input audio signal and output MEG signal for all the five candidate models

(Source by Author)

**Task 2.2**

In this task, the model residual (error) sum of squares errors(RSS) is calculated for all 5 candidate models. RSS is calculated on the basis of estimated model parameters.

The mathematical expression of RSS is

where xi represents ith row in the input matrix X and is a column vector

The calculated RSS for all candidate models is set in *Table 1* below.

|  |  |
| --- | --- |
|  | RSS |
| Model 1 | 12025.421 |
| Model 2 | 11438.94 |
| Model 3 | 5128.312 |
| Model 4 | 2102.068 |
| Model 5 | 1836.1676 |

Table : Calculated RSS for all candidate models

**Task 2.3**

In this task, log-likelihood function for every candidate model is computed. The log-likelihood of a model can be expressed in a mathematical expression below.

In this equation, is the variance of model’s residual and can be expressed as

and n is total number of samples.

The log-likelihood function for all candidate models is plotted in *Figure 23* Log-likelihood of all candidate models below.

|  |  |  |
| --- | --- | --- |
| Chart, line chart  Description automatically generated  Model 1 | Chart, line chart  Description automatically generated  Model 2 | Chart  Description automatically generated  Model 3 |

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated  Model 4 | Chart, line chart  Description automatically generated  Model 5 |

Figure : Log-likelihood of all candidate models

(Source by Author)

**Task 2.4**

In this task Akaike information criterion (AIC) and Bayesian information criterion (BIC) for every candidate model is computer using the formulas given below.

In the above expressions, k is the estimated parameters in each model and ) is the log-likelihood functions.

The calculated AIC and BIC for all candidate models is set in Table 3 below.

|  |  |  |
| --- | --- | --- |
|  | AIC | BIC |
| Model 1 | 1391.5133 | 1404.7065 |
| Model 2 | 1379.3400 | 1389.2350 |
| Model 3 | 1218.4623 | 1234.9538 |
| Model 4 | 1030.0507 | 1049.8407 |
| Model 5 | 997.93433 | 1014.4259 |

Table : AIC and BIC for all candidate models

**Task 2.5 and 2.6**

The best regression model according to the AIC, BIC and distribution of model residuals (RSS) from the set of 5 candidate models is model 5. It is because for the model to perform best, the sum of squares errors (RSS) should be as less as possible. Moreover, the AIC and BIC should be as less as possible than other candidate models. RSS, AIC and BIC for model 5 is the least ones. Therefore, model 5 is the best model and using this model further training and testing is performed and the data is split into 70% for training and 30% for testing.

**Task 2.7**

In this task, the data is split into 70% for training and 30% for testing. Since, the best model is model 5, the model parameter estimation is done using the training sample data and output MEG signals is predicted using the testing sample data. The regression plot for testing input and output signal is shown in *Figure 24*.

Chart, scatter chart

Description automatically generated

Figure : Regression plot for input audio signal and output MEG signal

(Source by Author)

The 95% model prediction confidence interval are calculated and is plotted with model prediction in *Figure 25*. The 95% confidence interval is 0.6068 and 0.9705 according to the distribution plot.

Chart, histogram

Description automatically generated

Figure : 95% confidence interval distribution plot

(Source by Author)

**Task 3: Approximate Bayesian Computation (ABC)**

Approximate Bayesian Computation (ABC) is method for determining the posterior distribution by ignoring the calculation of a likelihood function (Leyshon, 2021). The ABC algorithm uses the samples from the prior distribution to estimate the posterior of the parameter by simulating the model to produce artificial datasets X. In ABC method, the comparison of how close the artificial dataset X is produced by the sample parameter from the observed real dataset Y is computer using ρ(X, Y). The produced artificial datasets X which are very close to the real dataset Y is then collected as samples for posterior distribution. The closeness ‘very close’ is basically determined by a pre-defined parameter called tolerance ε. In this project, we will use the ABC based on the Sequential Monte Carlo (SMC) for evaluating the parameters. AMC SMC helps in determining the sensitivity and unpredictability of the parameters and make changes so that it enhances the performance of the model than ABC. Here, we will develop ABC SMC for sample selection by applying several mathematical interpretations. ABC SMC is able to select he best model using the standard Poison model selection tool. The rejection Approximate Bayesian Computation method to compute the distribution of model parameters of model 5. The value for θ1 and θbias is taken from the candidate model 5 making all other θ values constant. The rejection ABC algorithm is used to find the posterior distribution on the selected model 5 and likelihood for input distribution is shown in *Figure 26*.

Chart, histogram

Description automatically generated

Figure : Likelihood distribution for input audio data

(Source by Author)

The posterior and prior distribution for the input audio data on model 5 is depicted in *Figure 27* below.

Chart, line chart

Description automatically generated

Figure : The posterior and prior distribution for the input audio data on model 5

(Source by Author)

The posterior and prior distribution for an input audio signal on model 5 with approximate posterior and ground truth is used for splitting the distribution. These distributions are plotted on the basis of different epsilon and acceptance ratio values in *Figure 28, Figure 29, Figure 30* and *Figure 31.*

|  |  |
| --- | --- |
| **Chart, histogram  Description automatically generated**  Figure : epsilon = 0.5, acceptance ratio = 0.0001  (Source by Author) | **Chart, line chart, histogram  Description automatically generated**  Figure : epsilon = 0.4, acceptance ratio = 0.0001  (Source by Author) |
| **Chart, line chart, histogram  Description automatically generated**  Figure : epsilon = 0.3, acceptance ratio = 0.0001  (Source by Author) | **Chart, line chart  Description automatically generated**  Figure : epsilon = 0.2, acceptance ratio = 0.0001  (Source by Author) |

1. **Conclusion**

In this project, modelling of brain signals using nonlinear regression is performed. The brain signal dataset having neutral and emotional audio signals has been analyzed and predicted using nonlinear regression technique. The initial exploratory data analysis of the dataset was performed using python programming language in jupyter notebook environment. The series of tasks were performed for investigating and plotting distributions of input audio signal and output MEG sound signal in Task 1. In Task 2 we identified the true model from 5 candidate models for analyzing the relationship between input audio and output brain signal. From the set of 5 candidate models, the best model to truly define the relationship between input and output was model 5. The model 5 was selected using the model residual (error) and squared errors (RSS), AIC and BIC. For the model to be best, the sum of squares errors (RSS) should be as less as possible. The AIC and BIC should be as less as possible than other candidate models, also the AIC should be lower than the BIC. All the data was used for training during model prediction and model 5 was trained by splitting the data into 70% for training and 30% for testing of the input audio data. The 95% confidence was computed. Finally, in Task 3 the Approximate Bayesian Computation (ABC) was used to plot the prior and posterior distribution for data on model 5.

1. **References**

Kenton, W. (2022). *What Is Nonlinear Regression? Comparison to Linear Regression* [online] available from <<https://www.investopedia.com/terms/n/nonlinear-regression.asp#:~:text=Nonlinear%20regression%20is%20a%20form,a%20nonlinear%20(curved)%20relationship>> [27 November 2022]

Singh, S.P. (2014). ‘Magnetoencephalography: basic principles’. *Annals of Indian Academy of Neurology*, *17*(Suppl 1), p.S107. available from < <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4001219/>> [27 November 2022]

Leyshon, T. (2021). *The ABCs of Approximate Bayesian Computation* [online] available from <<https://towardsdatascience.com/the-abcs-of-approximate-bayesian-computation-bfe11b8ca341>> [27 November 2022]

**Appendix**

**Github Link:** <https://github.com/guddu095/Modelling-brain-signals-using-nonlinear-regression.git>

**Codes**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import scipy

import scipy.stats

import statsmodels.api as sm

from sklearn.utils import resample

from sklearn.metrics import accuracy\_score

from scipy.stats import norm, invgamma, gaussian\_kde

from matplotlib import pyplot as plt

from cycler import cycler

np.random.seed(123457)

%matplotlib inline

X = pd.read\_csv("X.csv")

y = pd.read\_csv("y.csv")

time = pd.read\_csv("time.csv")

X.head()

y.head()

time.head()

x1 = list(X['x1'])

x2 = list(X['x2'])

y1 = list(y['y'])

time1 = list(time['time'])

df = pd.DataFrame({

'time' : time1,

'x1' : x1,

'x2' : x2,

'y' : y1

})df

**Task 1**

# Time series plots (of input audio and output MEG signal)

plt.plot(df['time'],df['x1'], marker = 'o')

plt.xlabel("time")

plt.ylabel("input audio")

plt.title("Time Series Plot for input")

plt.plot(df['time'],df['y'], marker = 'o')

plt.xlabel("time")

plt.ylabel("output MEG signal")

plt.title("Time Series Plot for output")

# Distribution for each (input & output) signal

sns.displot(df, x="x1")

plt.xlabel("Input audio")

plt.ylabel("Count")

plt.title("Input Signal Distribution")

plt.show()

sns.displot(df, x="y")

plt.xlabel("Output MEG")

plt.ylabel("Count")

plt.title("Output MEG Signal Distribution")

plt.show()

plt.scatter(df['x1'],df["y"], marker='o')

# Labelling

plt.xlabel("input audio")

plt.ylabel("output MEG signal")

plt.title("Scatter plot")

plt.show()

names=df.columns

correlation\_mat = df.corr()

sns.heatmap(correlation\_mat, annot = True)

plt.title("Correlation matrix")

plt.xlabel("Features")

plt.ylabel("Features")

plt.show()

# Boxplots of output brain signals to examine effect of sound categories

fig = plt.figure(figsize =(10, 7))

# Creating plot

plt.boxplot(df['y'])

# show plot

plt.show()

# Preliminary data analysis for each type of input sound signal separately (i.e. when 𝑥2 = 0, and 𝑥2 = 1)

# Neutral Audio Preliminary Analysis

df\_x\_0 = df[df['x2'] ==0]

df\_x\_0.head()

plt.plot(df\_x\_0['time'],df\_x\_0["x1"], marker='o')

# Labelling

plt.xlabel("Time")

plt.ylabel("Input audio")

plt.title("Time series plot when x2=0")

# Display

plt.show()

plt.plot(df\_x\_0['time'],df\_x\_0["y"], marker='o')

# Labelling

plt.xlabel("Time")

plt.ylabel("output MEG signal")

plt.title("Time series plot when x2=0")

# Display

plt.show()

sns.displot(df\_x\_0, x="x1")

plt.xlabel("Input audio")

plt.ylabel("Count")

plt.title("Distribution input signal when x2=0")

# Display

plt.show()

plt.scatter(df\_x\_0['x1'],df\_x\_0["y"], marker='o')

# Labelling

plt.xlabel("Input audio")

plt.ylabel("output MEG signal")

plt.title("Scatter plot when x2=0")

# Display

plt.show()

correlation\_mat = df\_x\_0.corr()

sns.heatmap(correlation\_mat, annot = True)

plt.title("Correlation matrix when x2=0")

plt.xlabel("Features")

plt.ylabel("Features")

plt.show()

fig = plt.figure(figsize =(10, 7))

# Creating plot

plt.boxplot(df\_x\_0['y'])

plt.title("Box plot for Output MEG when x2=0")

# show plot

plt.show()

sns.displot(df\_x\_0, x="y")

plt.xlabel("Output MEG")

plt.ylabel("Count")

plt.title("Distribution Output MEG when x2=0")

# Display

plt.show()

# Emotional Audio Preliminary Analysis

df\_x\_1 = df[df['x2'] ==1]

df\_x\_1.head()

plt.plot(df\_x\_1['time'],df\_x\_1["x1"], marker='o')

# Labelling

plt.xlabel("Time")

plt.ylabel("Input audio")

plt.title("Time series plot when x2=1")

# Display

plt.show()

plt.plot(df\_x\_1['time'],df\_x\_1["y"], marker='o')

# Labelling

plt.xlabel("Time")

plt.ylabel("output MEG signal")

plt.title("Time series plot when x2=1")

# Display

plt.show()

sns.displot(df\_x\_1, x="x1")

plt.xlabel("Input audio")

plt.ylabel("Count")

plt.title("Distribution input signal when x2=1")

# Display

plt.show()

sns.displot(df\_x\_1, x="y")

plt.xlabel("Output MEG")

plt.ylabel("Count")

plt.title("Distribution Output MEG when x2=1")

# Display

plt.show()

plt.scatter(df\_x\_1['x1'],df\_x\_1["y"], marker='o')

# Labelling

plt.xlabel("Input audio")

plt.ylabel("output MEG signal")

plt.title("Scatter plot when x2=1")

# Display

plt.show()

correlation\_mat = df\_x\_1.corr()

sns.heatmap(correlation\_mat, annot = True)

plt.title("Correlation matrix when x2=1")

plt.xlabel("Features")

plt.ylabel("Features")

plt.show()

fig = plt.figure(figsize =(10, 7))

# Creating plot

plt.boxplot(df\_x\_1['y'])

plt.title("Box plot for Output MEG when x2=1")

# show plot

plt.show()

**Task 2**

**Task 2.1**

# Model 1

X1=list(df['x1'])

X2=list(df['x2'])

X1\_3=[]

X1\_5=[]

for i in range(len(X1)):

X1\_3.append(pow(X1[i],3))

X1\_5.append(pow(X1[i],5))

X\_mod1 = np.vstack([X1\_3,X1\_5,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1\_1=df['y']

y1\_1 = y1\_1[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X\_mod1.T,X\_mod1)),X\_mod1.T)),y1\_1)

print(Theta)

y\_pred\_1=[]

for i in range(len(X1)):

y\_pred\_1.append(Theta[0]\*X1\_3[i] + Theta[1]\*X1\_5[i]+Theta[2]\*X2[i] + Theta[3]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1\_1, 'g.')

plt.plot(X1, y\_pred\_1, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

# Model 2

# assemble matrix A

X1=list(df['x1'])

X2=list(df['x2'])

X\_mod2 = np.vstack([X1,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1\_2=df['y']

y1\_2 = y1\_2[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X\_mod2.T,X\_mod2)),X\_mod2.T)),y1\_2)

print(Theta)

y\_pred\_2=[]

for i in range(len(X1)):

y\_pred\_2.append(Theta[0]\*X1[i] + Theta[1]\*X2[i]+ Theta[2]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1\_2, 'g.')

plt.plot(X1, y\_pred\_2, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

# Model 3

# assemble matrix A

X1=list(df['x1'])

X2=list(df['x2'])

X1\_3=[]

X1\_4=[]

for i in range(len(X1)):

X1\_3.append(pow(X1[i],3))

X1\_4.append(pow(X1[i],4))

X\_mod3 = np.vstack([X1,X1\_3,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1\_3=df['y']

y1\_3 = y1\_3[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X\_mod3.T,X\_mod3)),X\_mod3.T)),y1\_3)

print(Theta)

y\_pred\_3=[]

for i in range(len(X1)):

y\_pred\_3.append(Theta[0]\*X1[i] + Theta[1]\*X1\_3[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1\_3, 'g.')

plt.plot(X1, y\_pred\_3, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

# Model 4

# assemble matrix A

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X\_mod4 = np.vstack([X1,X1\_2,X1\_3,X1\_5,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1\_4=df['y']

y1\_4 = y1\_4[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X\_mod4.T,X\_mod4)),X\_mod4.T)),y1\_4)

print(Theta)

y\_pred\_4=[]

for i in range(len(X1)):

y\_pred\_4.append(Theta[0]\*X1[i] + Theta[1]\*X1\_2[i]+ Theta[2]\*X1\_3[i]+Theta[3]\*X1\_5[i]+Theta[4]\*X2[i]+Theta[5]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1\_4, 'g.')

plt.plot(X1, y\_pred\_4, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

# Model 5

# assemble matrix A

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X\_mod5 = np.vstack([X1,X1\_2,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1\_5=df['y']

y1\_5 = y1\_5[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X\_mod5.T,X\_mod5)),X\_mod5.T)),y1\_5)

print(Theta)

y\_pred\_5=[]

for i in range(len(X1)):

y\_pred\_5.append(Theta[0]\*X1[i] + Theta[1]\*X1\_2[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1\_5, 'g.')

plt.plot(X1, y\_pred\_5, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

**Task 2.2**

RSS\_model1 = (np.sum(np.square(y1\_1-y\_pred\_1)))

RSS\_model1

RSS\_model2 = (np.sum(np.square(y1\_2-y\_pred\_2)))

RSS\_model2

RSS\_model3 = (np.sum(np.square(y1\_3-y\_pred\_3)))

RSS\_model3

RSS\_model4 = (np.sum(np.square(y1\_4-y\_pred\_4)))

RSS\_model4

RSS\_model5 = (np.sum(np.square(y1\_5-y\_pred\_5)))

RSS\_model5

**Task 2.3**

# Model 1

y2\_1 = []

for i in range(len(y\_pred\_1)):

y2\_1.append(np.log(scipy.stats.norm.pdf(y\_pred\_1,i,1)).sum())

plt.plot(y\_pred\_1,y2\_1)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

X\_mod1 = sm.add\_constant(X\_mod1)

#fit regression model

model\_1 = sm.OLS(y1\_1, X\_mod1).fit()

# Model 2

y2\_2 = []

for i in range(len(y\_pred\_2)):

y2\_2.append(np.log(scipy.stats.norm.pdf(y\_pred\_2,i,1)).sum())

plt.plot(y\_pred\_2,y2\_2)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

import statsmodels.api as sm

X\_mod2 = sm.add\_constant(X\_mod2)

#fit regression model

model\_2 = sm.OLS(y1\_2, X\_mod2).fit()

# Model 3

y2\_3 = []

for i in range(len(y\_pred\_3)):

y2\_3.append(np.log(scipy.stats.norm.pdf(y\_pred\_3,i,1)).sum())

plt.plot(y\_pred\_3,y2\_3)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

X\_mod3 = sm.add\_constant(X\_mod3)

#fit regression model

model\_3 = sm.OLS(y1\_3, X\_mod3).fit()

# Model 4

y2\_4 = []

for i in range(len(y\_pred\_4)):

y2\_4.append(np.log(scipy.stats.norm.pdf(y\_pred\_4,i,1)).sum())

plt.plot(y\_pred\_4,y2\_4)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

X\_mod4 = sm.add\_constant(X\_mod4)

#fit regression model

model\_4 = sm.OLS(y1\_4, X\_mod4).fit()

# Model 5

y2\_5 = []

for i in range(len(y\_pred\_5)):

y2\_5.append(np.log(scipy.stats.norm.pdf(y\_pred\_5,i,1)).sum())

plt.plot(y\_pred\_5,y2\_5)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

X\_mod5 = sm.add\_constant(X\_mod5)

#fit regression model

model\_5 = sm.OLS(y1\_5, X\_mod5).fit()

**Task 2.4**

# Model 1

#view AIC of model

print("AIC : ",model\_1.aic)

#view BIC of model

print("BIC : ",model\_1.bic)

# Model 2

#view AIC of model

print("AIC : ",model\_2.aic)

#view BIC of model

print("BIC : ",model\_2.bic)

# Model 3

#view AIC of model

print("AIC : ",model\_3.aic)

#view BIC of model

print("BIC : ",model\_3.bic)

# Model 4

#view AIC of model

print("AIC : ",model\_4.aic)

#view BIC of model

print("BIC : ",model\_4.bic)

# Model 5

#view AIC of model

print("AIC : ",model\_5.aic)

#view BIC of model

print("BIC : ",model\_5.bic)

# Best model candidate is model 5

# assemble matrix A

df1=df.iloc[0:160,:]

X1=list(df1['x1'])

X2=list(df1['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X = np.vstack([X1,X1\_2,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1=[]

y1=df1['y']

y1 = y1[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X.T,X)),X.T)),y1)

print(Theta)

y\_pred=[]

for i in range(len(X1)):

y\_pred.append(Theta[0]\*X1[i] + Theta[1]\*X1\_2[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1, 'g.')

plt.plot(X1, y\_pred, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

RSS\_model = (np.sum(np.square(y1-y\_pred)))

RSS\_model

y2 = []

for i in range(len(y\_pred)):

y2.append(np.log(scipy.stats.norm.pdf(y\_pred,i,1)).sum())

plt.plot(y\_pred,y2)

plt.title(r'Log-Likelihood')

plt.xlabel(r'$\mu$')

plt.grid()

plt.savefig("likelihood\_normal\_distribution\_02.png", bbox\_inches='tight')

#plt.show()

X = sm.add\_constant(X)

#fit regression model

model = sm.OLS(y1, X).fit()

#view AIC of model

print("AIC : ",model.aic)

#view BIC of model

print("BIC : ",model.bic)

# assemble matrix A

df1=df.iloc[160:200,:]

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X = np.vstack([X1,X1\_2,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1=df['y']

y1 = y1[:, np.newaxis]

y\_pred=[]

for i in range(len(X1)):

y\_pred.append(Theta[0]\*X1[i] + Theta[1]\*X1\_2[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1, 'g.')

plt.plot(X1, y\_pred, 'y.')

plt.xlabel('x')

plt.ylabel(' y')

plt.show()

**Task 2.7**

# Training

df1=df.iloc[0:140,:]

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X = np.vstack([X1,X1\_2,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1=df['y']

y1 = y1[:, np.newaxis]

# Theta = np.dot((np.dot(np.linalg.inv(np.dot(X.T,X)),X.T)),y1)

# print(Theta)

y\_pred=[]

for i in range(len(X1)):

y\_pred.append(Theta[0]\*X1[i] + Theta[1]\*X1\_2[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1, 'g.')

plt.plot(X1, y\_pred, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

#Testing

# assemble matrix A

df1=df.iloc[140:200,:]

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X = np.vstack([X1,X1\_2,X1\_4,X2,np.ones(len(X2))]).T

# turn y into a column vector

y1=df['y']

y1 = y1[:, np.newaxis]

y\_pred=[]

for i in range(len(X1)):

y\_pred.append(Theta[0]\*X1[i] + Theta[1]\*X1\_3[i]+Theta[2]\*X1\_4[i] + Theta[3]\*X2[i]+Theta[4]+1)

# plot the results

plt.figure(figsize = (10,8))

plt.plot(X1, y1, 'g.')

plt.plot(X1, y\_pred, 'y.')

plt.xlabel('x')

plt.ylabel('y')

plt.show()

x =X

# configure bootstrap

n\_iterations = 1000 # here k=no. of bootstrapped samples

n\_size = int(len(x))

# run bootstrap

medians = list()

for i in range(n\_iterations):

s = resample(x, n\_samples=n\_size);

m = np.median(s);

medians.append(m)

# plot scores

plt.hist(medians)

plt.show()

# confidence intervals

alpha = 0.95

p = ((1.0-alpha)/2.0) \* 100

lower = np.percentile(medians, p)

p = (alpha+((1.0-alpha)/2.0)) \* 100

upper = np.percentile(medians, p)

print(f"\n{alpha\*100} confidence interval {lower} and {upper}")

**Task 3**

# assemble matrix A

df1=df.iloc[0:160,:]

X1=list(df['x1'])

X2=list(df['x2'])

X1\_2=[]

X1\_4=[]

for i in range(len(X1)):

X1\_2.append(pow(X1[i],2))

X1\_4.append(pow(X1[i],4))

X = np.vstack([X1,np.ones(len(X2))]).T

# turn y into a column vector

y1=df['y']

y1 = y1[:, np.newaxis]

# Direct least square regression

Theta = np.dot((np.dot(np.linalg.inv(np.dot(X.T,X)),X.T)),y1)

print(Theta)

y\_pred=[]

for i in range(len(X1)):

y\_pred.append(Theta[0]\*X1[i] + 6.2\*X1\_2[i]+(-0.285)\*X1\_4[i] + 4.34\*X2[i]+Theta[1]+1)

# import numpy as np

# from scipy.stats import norm, invgamma, gaussian\_kde

# from matplotlib import pyplot as plt

# from cycler import cycler

# np.random.seed(123457)

# %matplotlib inline

plt.rcParams.update({'lines.linewidth': 2})

colors=[plt.cm.Set2(i) for i in np.linspace(0, 1, 8)]

plt.rcParams.update({'axes.prop\_cycle': cycler('color', colors)})

groundtruth=2.

likelihood=norm(loc=0.,scale=np.sqrt(groundtruth))

(dmin,dmax)=(-5,5)

data=X1

lh\_data=likelihood.pdf(data)

x\_arr=X

f\_arr=likelihood.pdf(x\_arr)

plt.figure(figsize=(10,6))

plt.scatter(x\_arr,f\_arr,color='r',label="Likelihood")

markerline, stemlines, baseline = plt.stem(data,lh\_data,linefmt='-y',markerfmt='g.',label="Data")

baseline.set\_visible(False)

plt.title("Likelihood & Data")

plt.legend()

plt.ylim(bottom=0.)

plt.show()

alpha=60

beta=130

Nsamp=200

prior=invgamma(alpha,loc=0,scale=beta)

alphaprime=alpha+Nsamp/2

data2=[]

for i in range(len(data)):

data2.append(pow(data[i],2))

betaprime=beta+1/2.\*np.sum(data2)

posterior=invgamma(alphaprime,loc=0,scale=betaprime)

a=1

b=4

x=np.arange(a,b,0.01)

plt.figure(figsize=(10,6))

plt.xlim([a,b])

plt.xlabel("$\sigma^2$")

plt.ylim([0,1.2])

plt.plot([groundtruth,groundtruth],[0,1.2],linestyle='--',color='black',label="groundtruth")

plt.plot(x,prior.pdf(x)/prior.pdf(x).max(),label="prior")

plt.plot(x,posterior.pdf(x)/posterior.pdf(x).max(),label="posterior")

plt.legend()

plt.show()

def simulator(var):

return norm(loc=0.,scale=np.sqrt(var)).rvs(size=Nsamp)

def distance(sim\_ss,data\_ss):

return np.sqrt(np.sum((sim\_ss-data\_ss)\*\*2))

#Likelihood-free rejection sampling

def simulator(var):

return norm(loc=0.,scale=np.sqrt(var)).rvs(size=Nsamp)

def distance(sim\_ss,data\_ss):

return np.sqrt(np.sum((sim\_ss-data\_ss)\*\*2))

Ntries=10000

# sufficient summary statistics

def sufficient\_summary\_stat(data):

return np.var(data)

data\_ss=sufficient\_summary\_stat(data)

for epsilon in [0.5,0.4,0.3,0.2]:

# likelihood-free rejection sampler

# note that we never call the likelihood function!

samples=[]

for this\_try in range(Ntries):

this\_var = prior.rvs(size=1)

this\_sim = simulator(this\_var)

this\_ss = sufficient\_summary\_stat(this\_sim)

if(distance(this\_ss,data\_ss)<epsilon):

samples.append(this\_var)

samples=np.array(samples).T

fraction\_accepted=float(len(samples))/Ntries

# kernel density estimation of the approximate posterior

kernel=gaussian\_kde(samples)

# produce a plot

plt.figure(figsize=(10,6))

plt.xlim([a,b])

plt.xlabel("$\sigma^2$")

plt.ylim([0,1.2])

plt.plot([groundtruth,groundtruth],[0,1.2],linestyle='--',color='black',label="groundtruth")

plt.plot(x,prior.pdf(x)/prior.pdf(x).max(),label="prior")

plt.plot(x,posterior.pdf(x)/posterior.pdf(x).max(),label="true posterior")

plt.plot(x,kernel.evaluate(x)/kernel.evaluate(x).max(),label="approximate posterior")

plt.title("$\\varepsilon="+str(epsilon)+"$, acceptance ratio="+str(fraction\_accepted))

plt.legend()

plt.show()

# insufficient summary statistics: throw away most of the information

def insufficient\_summary\_stat(data):

return np.var(data[0:int(Nsamp/3.)])

data\_ss=insufficient\_summary\_stat(data)

for epsilon in [0.5,0.4,0.3,0.2,0.1]:

# likelihood-free rejection sampler

# note that we never call the likelihood function!

samples=[]

for this\_try in range(Ntries):

this\_var = prior.rvs(size=1)

this\_sim = simulator(this\_var)

this\_ss = insufficient\_summary\_stat(this\_sim)

if(distance(this\_ss,data\_ss)<epsilon):

samples.append(this\_var)

samples=np.array(samples).T[0]

fraction\_accepted=float(len(samples))/Ntries

# kernel density estimation of the approximate posterior

kernel=gaussian\_kde(samples)

# produce a plot

plt.figure(figsize=(10,6))

plt.xlim([a,b])

plt.xlabel("$\sigma^2$")

plt.ylim([0,1.2])

plt.plot([groundtruth,groundtruth],[0,1.2],linestyle='--',color='black',label="groundtruth")

plt.plot(x,prior.pdf(x)/prior.pdf(x).max(),label="prior")

plt.plot(x,posterior.pdf(x)/posterior.pdf(x).max(),label="true posterior")

plt.plot(x,kernel.evaluate(x)/kernel.evaluate(x).max(),label="approximate posterior")

plt.title("$\\varepsilon="+str(epsilon)+"$, acceptance ratio="+str(fraction\_accepted))

plt.legend()

plt.show()