FACULTY OF ENGINEERING, ENVIRONMENT & COMPUTING

(7089CEM)

Coursework
Introduction to Statistical methods for Data Science

MODULE LEADER: Dr Fei He
Student Name: Lekhasree Sripuram
SID: 10749896

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

Introduction

The purpose of the coursework is to find the best regression model among the given five nonlinear candidate models which can predict the output electroencephalogram (EEG) signal by taking 4 input EEG signals. In order to do so, the below tasks have been performed to study the relationship between different signals.

Datasets

The brain electroencephalogram (EEG) signals of csv files given for this task are X.csv, y.csv and time.csv. The X.csv file contains four input signals (x1, x2, x3, x4) whereas the y.csv and time.csv contains the output signal data and time in seconds data respectively each containing 201 rows.

Task 1: Preliminary data analysis

In this section, the following tasks such as time series plots, distribution of each signal and correlation between the signals are performed to understand the data.

1.1 Time series plots

Introduction

Time series plays a vital role in analysing the data when any variable vary with respect to time. It can be applied on any variable that gets changed over the time. As per the provided data, the four input signals are noted for every 0.002 seconds along with the output signal for each time interval.

In this step, using time series each EEG signal has plotted with the time to learn its relation.

Results

Below are the plots of time with input output signals. It shows the relationship between time and each signal. The code used to generate these plots can be referred in Appendix section code chunk-2

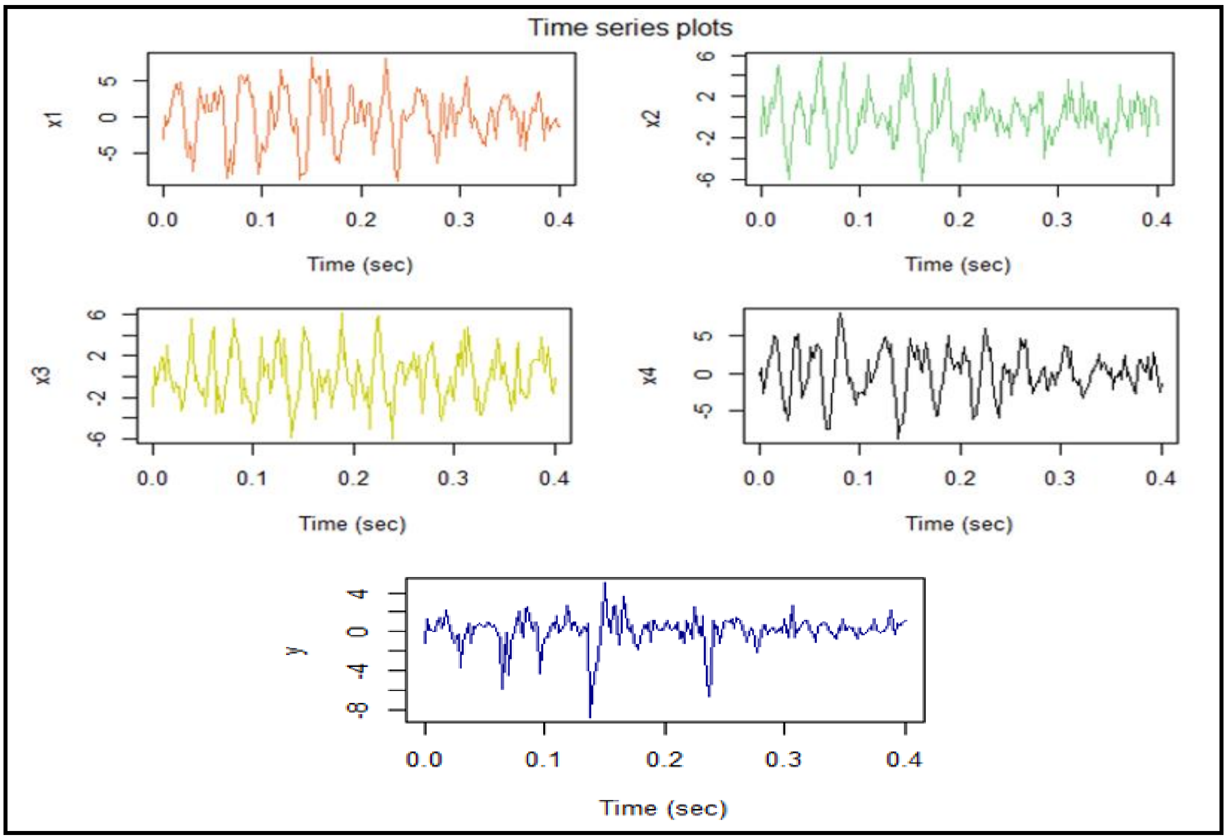

Figure 1 Time series plots of each signal

Discussion

The figure 1 and figure 2 shows how each signal vary over time. From the results, it can be noticed that the time on x-axis ranges from 0.0 to 0.4 seconds. All the signals are within the range of around -10 to 10 frequency. Also, all these five signals have additive noise. The above figure 1 is number of univariate plots that has plotted using plot() function whereas the below figure 2 is a multivariate plot that has plotted using ggplot() function. The univariate plots give how each signal acts over time. On the other hand, the multivariate plot shows the comparison of input four signals (x1, x2, x3, x4) along with output signal y. In addition to this, on seeing the above plots figure 1, it can be clearly known that this data has no null values.

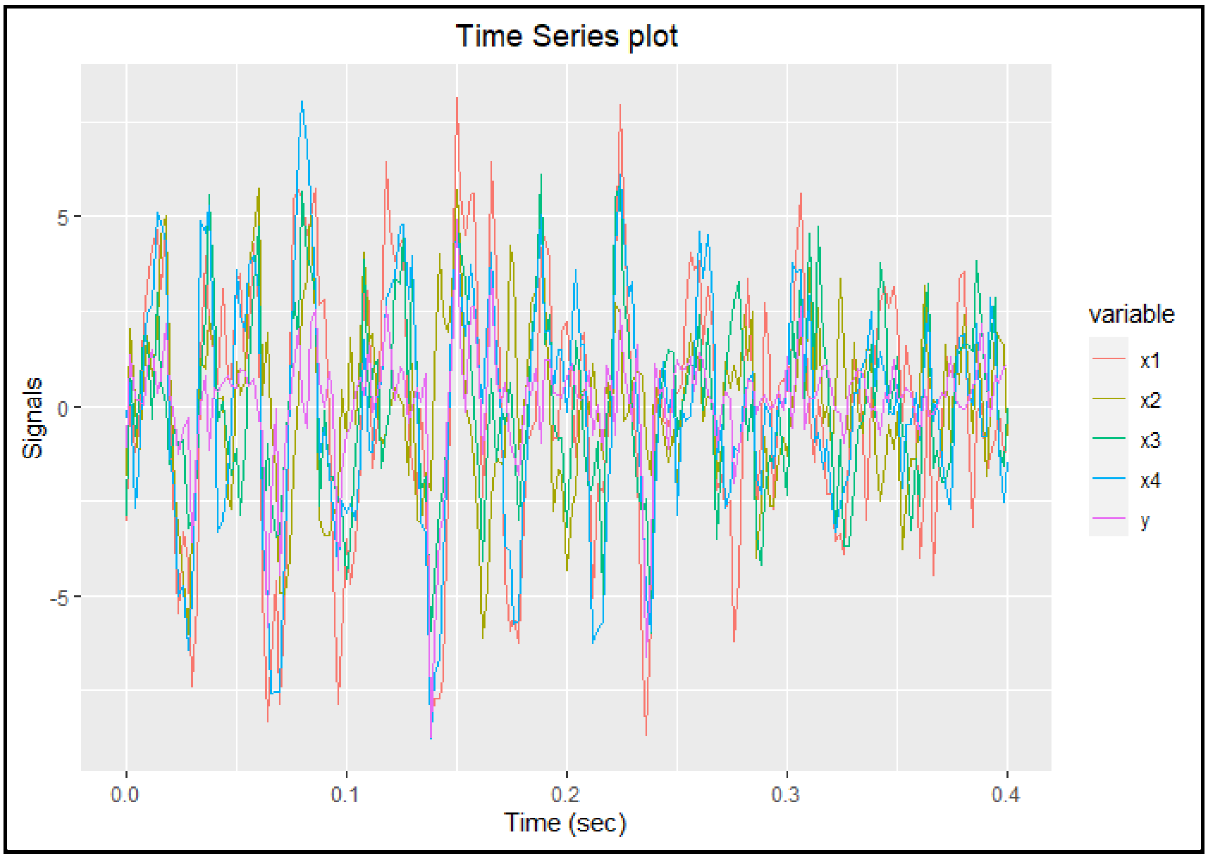

Figure 2 Time series plot of all signals

1.2 Distribution for each EEG signal

Introduction

In general, distributions are used to view the values of a variable in order to know how often each value occurs and the possibilities of the variable values. In this step, distributions have been plotted for each signal to understand the input signal values and its range of occurrence.

Results and Discussion

The distributions of each signal have plotted using hist plots. The code used for the plot has mentioned in the Appendix section code chunk-3

From the below figure 3, it can be noted that for the first input signal x1, there is a steady increase in the reoccurrences of values from -10 to 2 and the least repeated values range from around 7 to 10. Coming to the second input signal x2, the highest reoccurred value ranges from 0 to 1. On the other side, the highest reoccurred values for third input signal x3 ranges between -1 and 0. For the fourth input signal x4, the most reoccurred values are within 0 to 4 leading to less difference compared to the second highest reoccurred values -4 to 0. The output signal y has the highest reoccurred values between 0 and 1.

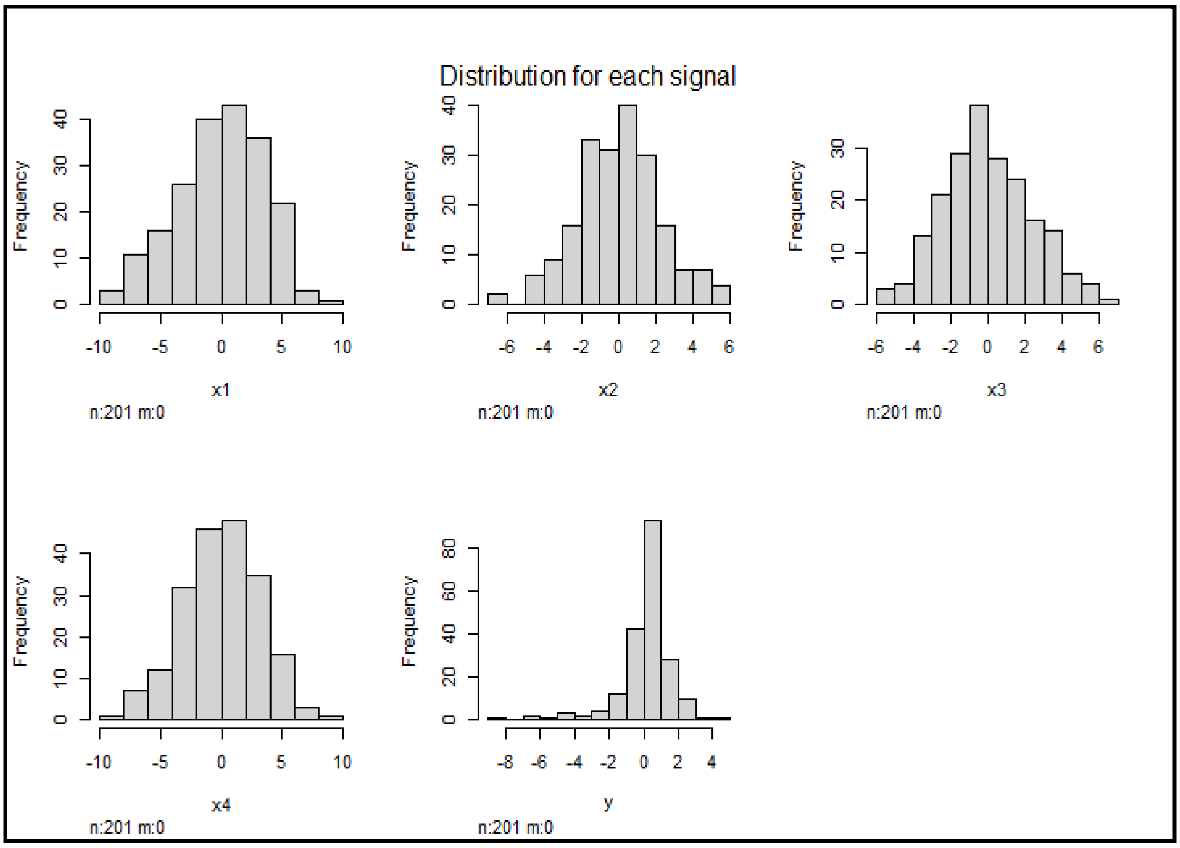

Figure 3 Distribution of each signal using hist plots

1.3 Correlation and scatter plots

Introduction

Correlation or dependency gives the statistical relationship of any two random variables. It shows how much one variable is related to another variable. The higher the value represents that the two variables are highly correlated whereas the lower the value represents that the two variables are less correlated. In this step, the correlation between different input signals and the output signal has shown to learn their dependencies.

Results

The below table and figure scatter plot shows the correlation between the variables one and another. It is done by using the code that shown in the Appendix section code chunk-4

Table 1 Correlation between the variables

	x1	**x2**	**x3**	**x4**	**y**
x1	1.0000000	0.2619758	0.5835224	0.8057686	0.7288988
x2	0.2619758	1.0000000	0.5207327	0.3658827	0.1663955
x3	0.5835224	0.5207327	1.0000000	0.7107202	0.4069755
x4	0.8057686	0.3658827	0.7107202	1.0000000	0.5705081
y	0.7288988	0.1663955	0.4069755	0.5705081	1.0000000

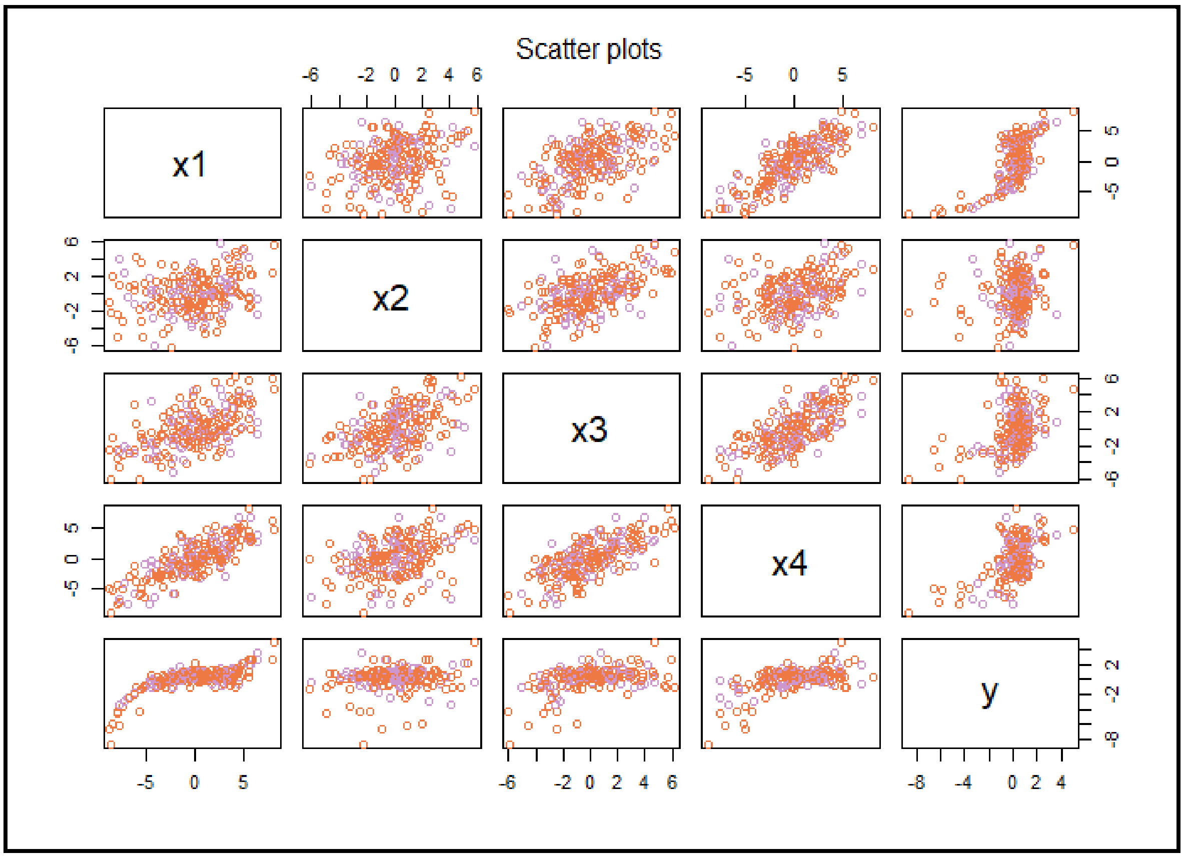

Figure 4 Scatter plot of correlation between variables

Discussion

On seeing the above scatter plot, it can be clearly said that all the signals are positively correlated. After noticing the scatter plots between few variables such as between x1 and x4, x3 and x4, x1 and y, it can be said that the correlation between them would be range from 0.7 to 0.8. The far the scatter points, represents the less correlated between the variables. It far scatter points can be noticed between x1 and x2, x2 and y.

For the accurate correlated values on observing the above table 1, the highest correlated value among these variables other than itself is between x1 and x4 input signals which is 0.8. The second highest correlated variables are input signal x1 and output signal y which have 0.728. The least correlated variables are input signal x2 and output signal y with 1.66.

Task 2: Regression – modelling the relationship between EEG signals

Introduction

The purpose of this section is to choose the best regression model among the given five candidate nonlinear polynomial regression models. It can be done by performing required measures and comparisons such as estimating model parameters, finding the error values, log-likelihood values, AIC and BIC values, and error distributions.

The five candidate models have been created as shown in Appendix section code chunk-5

2.1 Estimate model parameters

The parameter estimates are the coefficients that changes with a single unit change in predictor where other predictors are held in constant. These model parameters can be estimated by using least squares estimation method. Each coefficient defines the amount of contribution it did for a predictor.

Results and Discussion

Here, the model parameters are estimated for each candidate model using least squares as shown in Appendix section code chunk-6

Table 2 Estimate parameters of models

Estimate parameters of the candidate models				
Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
thetaHat1	**thetaHat1**	**thetaHat1**	**thetaHat1**	**thetaHat1**
-0.034101975	0.016334677	0.038109255	-0.034881772	-3.395313e-02
-0.001849575	-0.002713985	0.009827804	0.010482178	-2.701644e-04
0.010381813	0.303501144	-0.002092558	-0.001990496	1.034764e-02
-0.001949154		0.448299550	0.450329209	-1.943452e-03
0.468551685				-3.083194e-05
				4.606560e-01

The above table 2 shows the coefficients/estimate parameters of each candidate model. These parameters can be used for further measures which helps in comparing to decide the best regression model among the given five models.

2.2 Residual sum of squared errors

Residual sum of squared errors is a statistical method which is used to calculate the amount of variance of a dataset. It gives the error rate of the desired value with its predicted value. The less the RSS value, the better the model works in predicting the value and explaining the data. Here, the residual sum of squared errors has been applied to learn which model is better in predicting the value.

Results

The RSS values of each candidate model are calculated as shown in the Appendix section code chunk-7

Table 3 RSS values of the models

Models	**RSS values**
RSS1	57.32927
RSS2	351.4147
RSS3	57.32536
RSS4	57.46405
RSS5	57.30923

Discussion

The above table 3 gives the information on which candidate model has performed better in predicting by minimizing its error value. After noticing the values, it can be clearly said that except model value, all the other four models performed better with less than 60 error values.

2.3 Log-likelihood function

In general, the likelihood function defines how good the model fits by using the estimated parameters. It can be done using logarithmic transformations known as log-likelihood function. The higher the value of log-likelihood function, the better the model fits in it. Here, the log-likelihood function has applied on each candidate model by passing the model with its RSS value to find which model has a better fit.

Results

The log-likelihood values of the given five models have fetched by using the code shown in Appendix section code chunk-8

Table 4 Log-likelihood values of models

Model	Likelihood
Likelihood1	-159.1313
Likelihood2	-341.3534
Likelihood3	-159.1244
Likelihood4	-159.3673
Likelihood5	-159.0962

Discussion

The above table 4 clearly shows that the candidate model 2 is not performing good for this data whereas the other candidate model's log-likelihood ranges within -159 values. On noticing the values, the model 5 shown better performance with less difference compared to model 3 followed by model 1. So far models 5,3,1 have shown better performance falling in the same range.

2.4 Akaike information criterion (AIC) and Bayesian information criterion (BIC)

Akaike information criterion (AIC)

The Akaike information criterion is a prediction error estimator and also estimates the statistical model's quality of the given datasets. It evaluates how good the model fits in the given data. These can be calculated by passing the count of generated estimate parameters along with its log-likelihood values. The less the AIC value, the better the candidate model fits in the data.

Bayesian information criterion (BIC)

The Bayesian information criterion that helps in choosing the better model among finite set of models. These can be calculated by passing the count of generated estimate parameters along with its log-likelihood values. The less the BIC value, the better the model fits in the data.

Results

The AIC and BIC values of the given five models have found by using the code shown in Appendix section code chunk-9

Table 5 AIC BIC values of the models

Model	AIC	BIC
aicbic1	328.2626	344.7791
aicbic2	688.7068	698.6168
aicbic3	326.2489	339.4621
aicbic4	330.1923	350.0121

aicbic5	326.7346	339.9478

Discussion

From the above table 5, it can be noticed that the model with the AIC and BIC values is candidate model 2 which can be said that it is not fit for the dataset. Among the other four models, the first model with lowest AIC and BIC values is candidate model 3 with 326.248 and 339.462 respectively. The second model with least AIC and BIC values is candidate model 5 with 326.73 and 339.947 where there is a minute difference compared to the first model. Therefore, the better fit model based on AIC and BIC values is the candidate model 3.

2.5 Error distribution plots

Error distributions provides information of the point prediction on how the error delta is. It gives the difference between the original value and the model predicted value. In this step, the found error distributions for each candidate model has plotted to know whether the models are normally distributed or not. In order to do so, the general formula for error distributions that has mentioned below has used.

$$\boxed{\text{Error} = \text{original\_y} - \text{predicted\_y} \quad \text{-- (1)}}$$

Results

The code used to plot the error distributions for each candidate model has shown in Appendix section code chunk-10

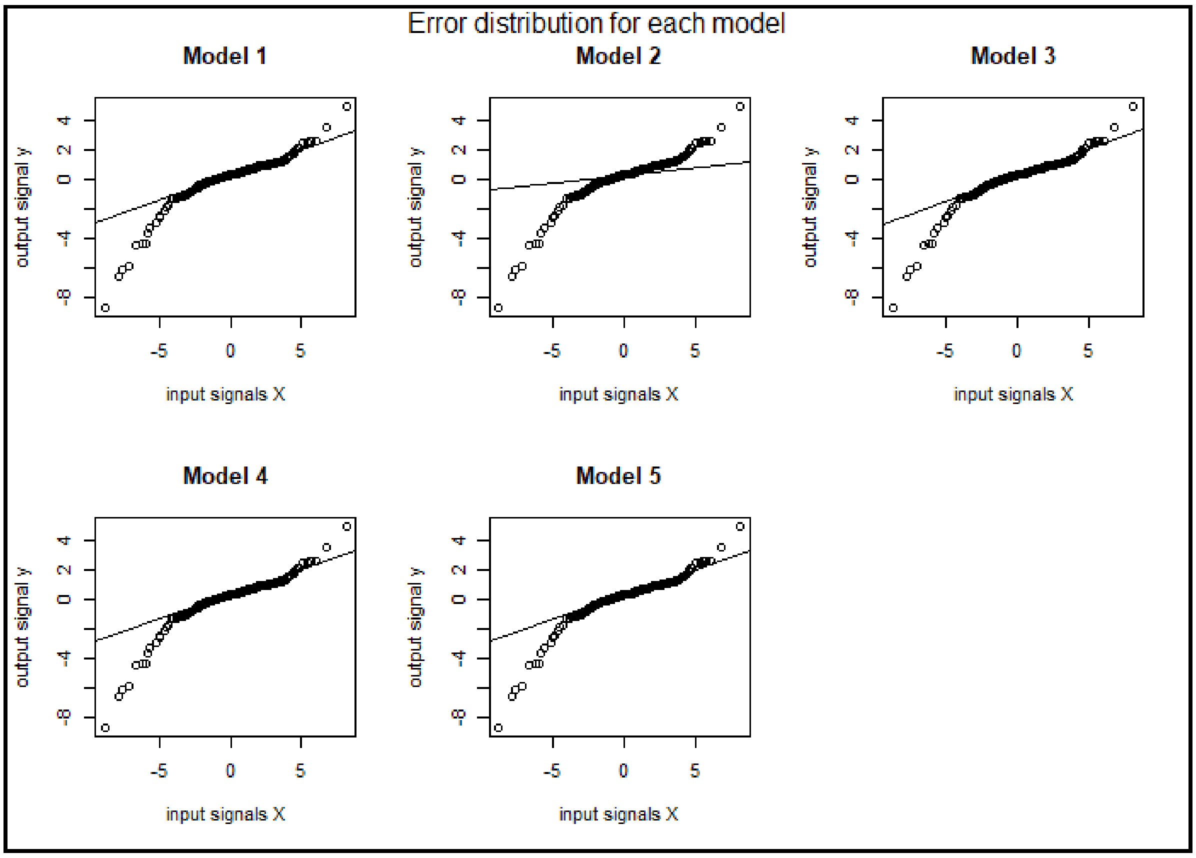

Figure 5 Error distributions

Discussion

The above figure 5 shows the error distribution plots for all the given five models. It can be done by using qqplot() function and qqline() function as well. The above error distributions were plotted between input signals(x) with output signal(y) and predicted output(yHat). It can be noticed that all the five models are not completely normally distributed. Among these five models, the model 2 is not at all normally distributed. Coming to the remaining four models, it can be noticed that they are similarly close to normal distributions. Yet, from the below figure 6 the model 3 the line that plotted with yellow colour, shows quite close to normal distribution compared to other models.

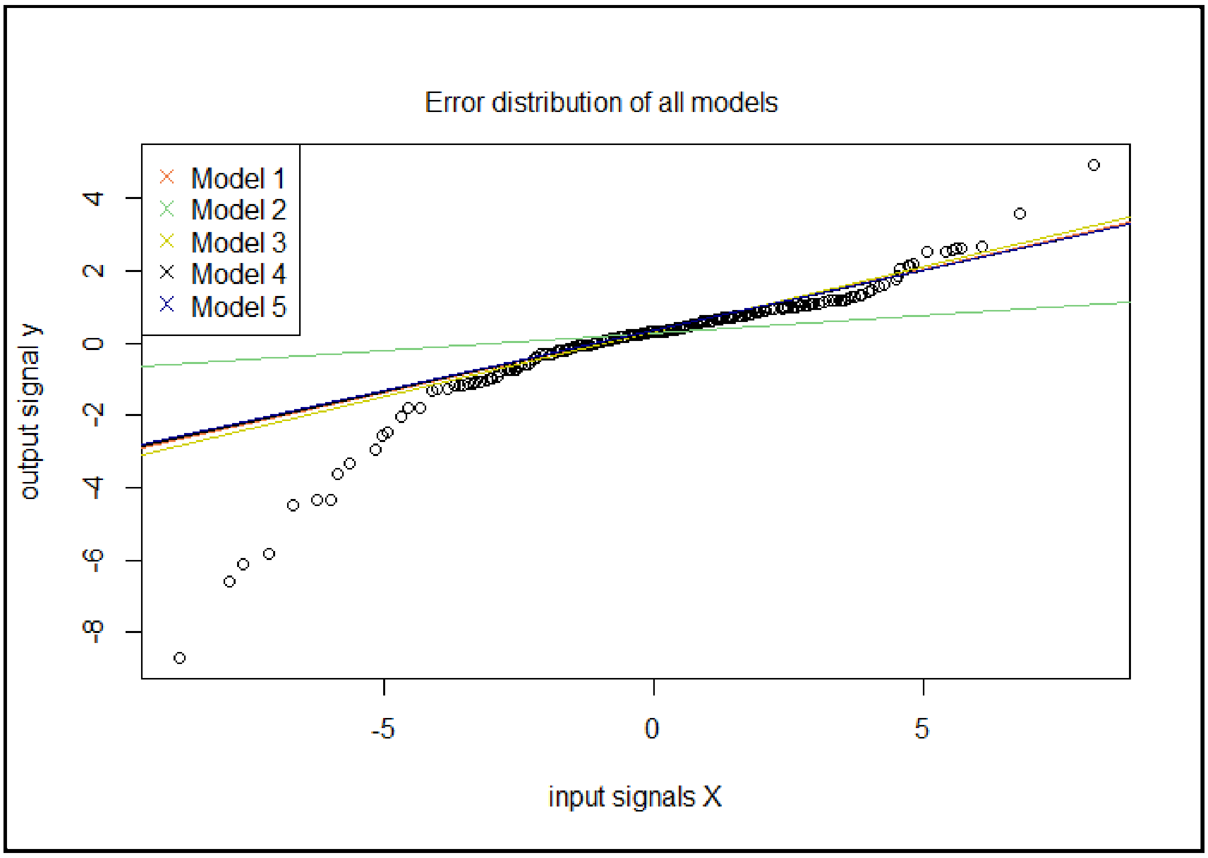

Figure 6 Error distributions of all models

2.6 Model Selection and Explanation

According to the above measures done on the given each candidate model, the candidate model 3 has shown its best performance n predicting the output signal. It has been decided on the below mentioned calculations.

Table 6 Calculations of all models

	RSS	**Log-likelihood**	**AIC**	**BIC**
Model 1	57.32927	-159.1313	328.2626	344.7791
Model 2	351.4147	-341.3534	688.7068	698.6168
Model 3	57.32536	-159.1244	326.2489	339.4621
Model 4	57.46405	-159.3673	330.1923	350.0121
Model 5	57.30923	-159.0962	326.7346	339.9478

The following discussion shows the comparisons of three models which performed well in each measure.

1. As per the residual sum of squared errors (RSS), the least value represents the better model. Therefore, from the above table 6, the model with least RSS value is candidate model 5 with 57.30923 followed by candidate model 3 with 57.32927 and candidate model 1 with 57.32927.
2. As per the log-likelihood, the model with the highest value represents as the better model. From the above table 6, the model with highest log likelihood value is candidate model 5 with -

159.0962 followed by candidate model 3 with -159.1244 and candidate model 1 with -159.1313.

3. The model with lowest AIC and BIC values represents as the best model. From the above table 6, the model with lowest AIC and BIC values is model 3 with 326.2489 and 339.4621 respectively.

From the above table 6, candidate model 2 has shown the least performance resulting that this model 2 is not well fit for this data. The remaining models have shown similar performance for this dataset. Though the candidate model 5 performed best in RSS and log-likelihood with less difference compared to model 3, the candidate model 3 has shown better results in AIC and BIC values. Also, from the figure 6 error distributions of all models, the candidate model 3 is close to normal distribution. Thus, the best regression model according to the calculations and observations done so far is the candidate model 3.

2.7 Model validation

To split the input and output values in order to use in training and testing purpose, the code that has shown in Appendix section code chunk-11 has been used. As given, the data has been split 70% for training and 30% for testing. After applying split on 201 rows, the train dataset has 141 rows whereas the test dataset has 60 rows.

The following tasks has been performed on the selected best model.

Estimate model parameters

To find the estimate model parameters of train data, the selected best model 3 has been used. The code behind finding the estimate parameters has shown in Appendix section code chunk-111

The following are the four estimate parameters of train data using the selected candidate model 3.

Table 7 Estimate model parameters of train data

Estimate model parameters
0.037102910
0.009833314
-0.002183413
0.455427158

Model's prediction on test data

The output signal values have been predicted by passing the above table 7 estimate parameters along with the selected candidate model 3. The code that has used to predict the output signals has mentioned in Appendix section code chunk-11

Confidence Interval and plotting error bars

The purpose of the step is to find the 95% confidence intervals and plot them with error bars with the predicted outputs of test data. In order to find the 95% confidence intervals, the below mentioned steps can be followed.

Before finding the confidence intervals, variance of the predicted values has to be calculated. This can be done by using the below formula

Var = sigma_square * (X %*% (solve(t(X)%*% X)) %*% t(X)) --- (1)

Confidence Interval = 1.96 * sqrt(Var) --- (2)

where X are the input test values, Var defines the variance of the predicted values and for the 95% confidence intervals, it has to be multiplied by 1.96.

Results and Discussion

The below figure shows the plot of the error bars by taking the test data input signals and the test data predicted values with in the range of 95% Confidence intervals. It has been plotted by using qqplot() function along with the segments() function with the range of 95% confidence intervals. The code that has used to plot this has shown in the Appendix section code chunk-11

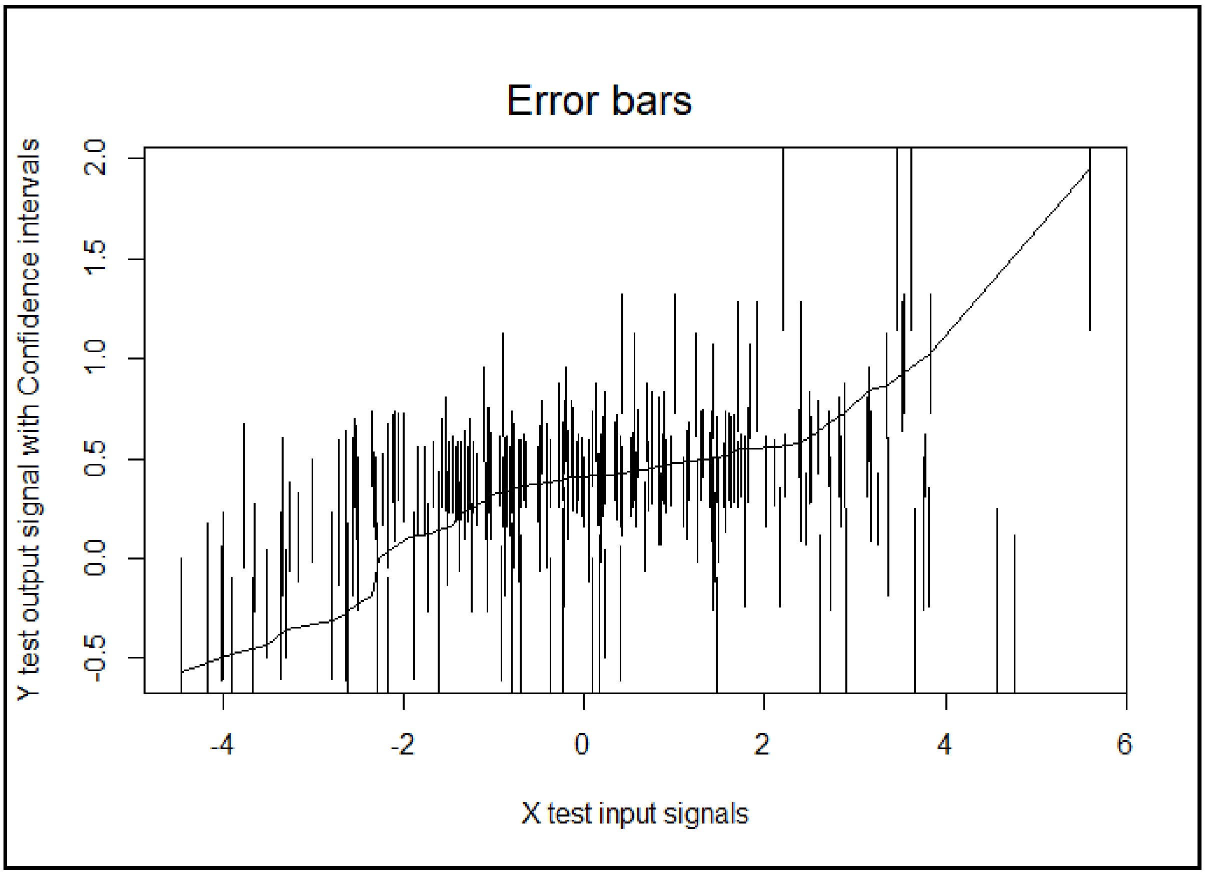

Figure 7 Error bars of test data

From the above figure 7, it can be noticed that there are few deviations from the normality at few points. Though there are deviations in middle, mostly the deviations have taken place at lower points and higher points.

Task 3 Approximate Bayesian Computation (ABC)

The purpose of this task steps is to compute the posterior distributions by using the rejection ABC method on selected candidate model 3.

Posterior distribution

The posterior distribution is a way to understand the uncertain quantities in the analysis after the data has been studied. It is the combination of likelihood function and prior distribution.

Rejection ABC method

The rejection ABC method use Monte Claro simulation in order to supply an alternative to likelihood. The following is the principle of this method.

- Creating a simulation model
- Specifying prior distribution for the estimate parameter
- Run the simulation
- If the simulated value is less than or equal to threshold, accept it. Otherwise reject it.
- The accepted values of the distribution acts as the posterior distribution.

Step 1: Choosing two largest value parameters for posterior distribution

In order to perform the method, two largest estimate parameters of the selected candidate model 3 have to be chosen. The below values are the estimate parameters of candidate model 3 taken from the table 2. The largest estimate parameters are 0.4482 and 0.03810 respectively.

Table 8 Estimate parameters of selected model

Model 3
0.038109255
0.009827804
-0.002092558
0.448299550

Step 2: Uniform distribution

After choosing the parameters, uniform distribution as prior has to be performed on these two parameters where random parameter values will be generated within the given range. It can be done by using runif() function. The code to perform the uniform distribution has shown in the Appendix section code chunk-13

Step 3: Rejection ABC

The rejection ABC method has to be applied on the samples generated from the uniform distribution. Here, among four estimate parameters, two were kept constant as the values got from the table 2 and replacing the other two largest parameters by each sample row. Then finding the predicted values of these estimated parameters with the candidate model 3. Later calculating the mean squared error of the predicted values by each sample row. If the mean squared error value is less than or equals to threshold, then accept the sample parameter value otherwise reject it.

The code used to perform this method has shown in the Appendix section code chunk-14

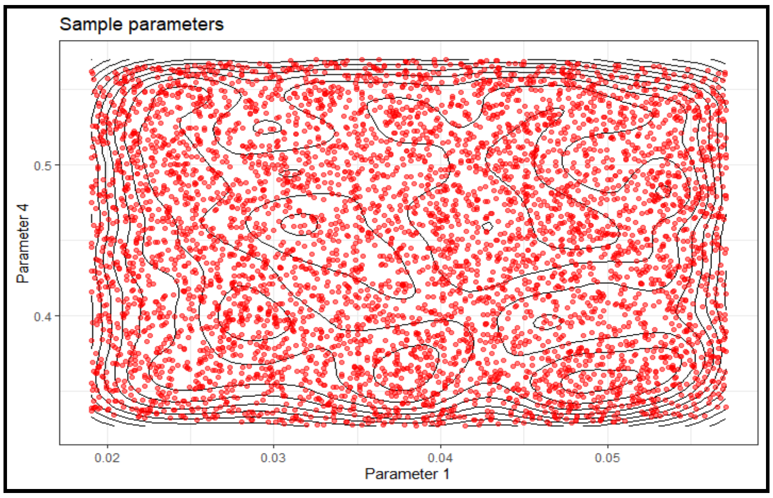

Figure 8 Sample theta parameters

Discussion

When the threshold value is 0.3, the total number of accepted values were 5190 which is half of the total parameters (10000). Therefore, the threshold value at 0.308 accepted 6400 parameters which prevents the risk of the low or high acceptance rates.

Step 4: Joint and marginal posterior distribution plots

The purpose of this step is to plot the joint posterior distribution and marginal posterior distribution of the sample parameters.

Joint posterior distribution

The joint posterior distribution shows how the relation is between the parameters and the variables. Here, the joint posterior distribution helps in understanding the relationship of sample parameters of the two largest parameters, where each sample parameter value of one theta falls with respect to other theta sample parameter.

Result

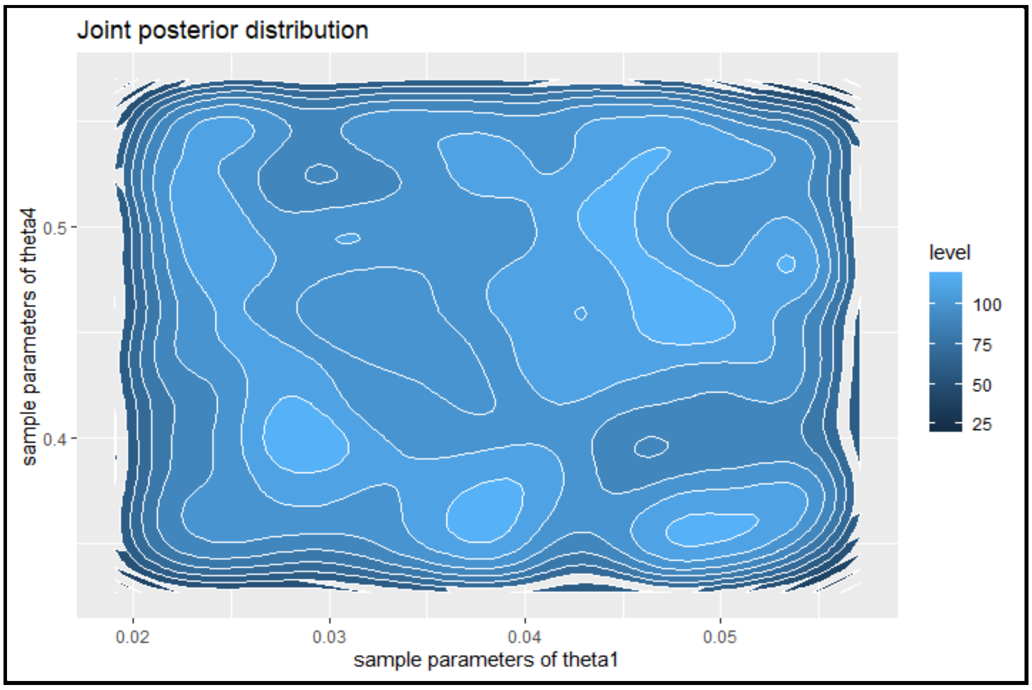

Figure 9 Joint posterior distribution

Discussion

The above figure 9 has been plotted by using ggplot() with contour 2D. It can be done by using stat_density_2d(). The code that has used to plot has mentioned in the Appendix section code chunk-16

Marginal posterior distribution

Marginal posterior distribution can be used to find the probability/ density distribution of the subset parameters. It shows the frequencies of the subset parameters within an interval. The below figure shows the marginal posterior distribution of the two parameter samples.

The code used for this result has shown in Appendix code chunk - 17

Result

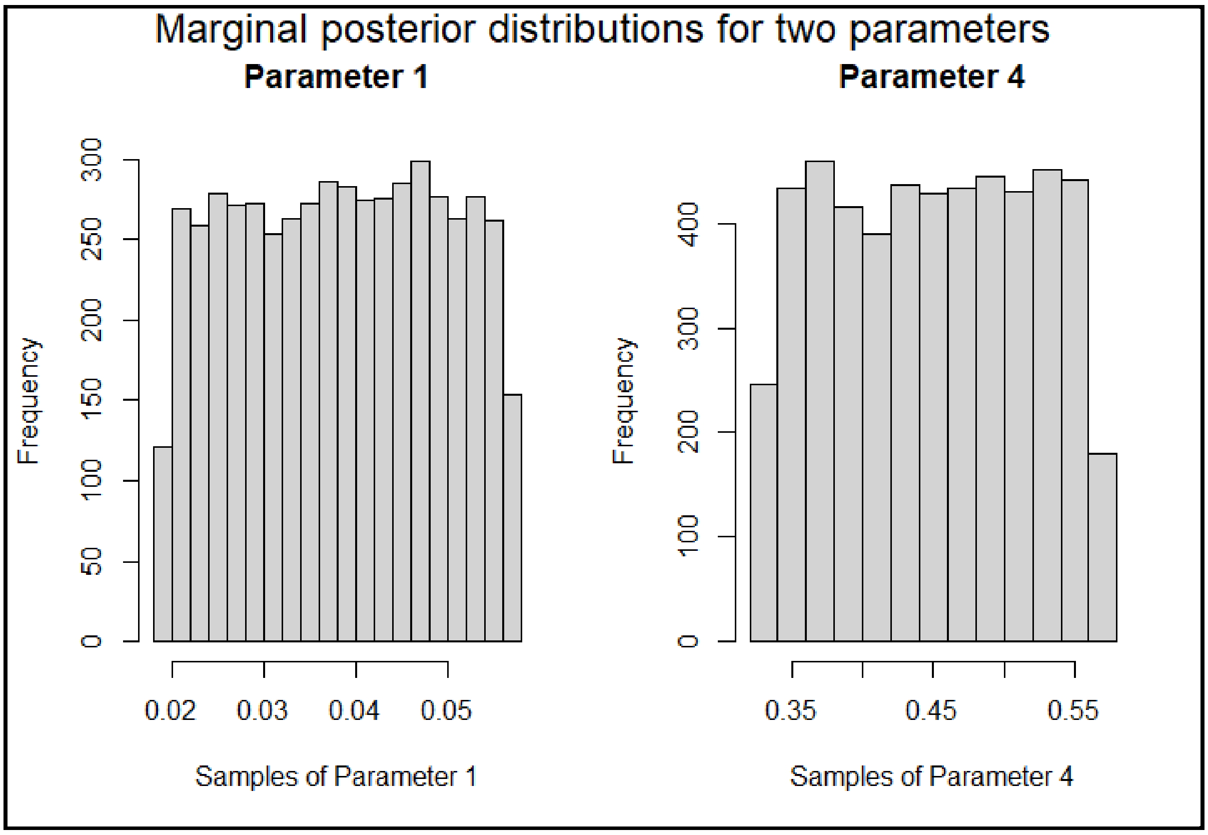

Figure 10 Marginal posterior distribution

3.5 Results and Discussions

The MSE value for the select model is 0.285, therefore choosing the epsilon value within the range. if epsilon is 0.3, we have 5190 samples, if 0.2, then 0 samples. As discussed above, 0.3 threshold value accepts only half of the parameters whereas 0.308 accepted 64% of each sample parameters list. From the figure 9, joint plot displays the relation between the sample parameters of theta1 and theta4 (two highest parameters). It reached the level almost 100 at the points 0.3 theta1 and 0.4 theta4. From the figure 10, marginal plot shows that sample parameters of theta1 ranging from around 120 to some fluctuations. On the other hand, samples of theta4 ranging with fluctuations between 180 to 500.

Conclusion

This paper has shown various calculations and measure to select the best regression model among the given five models which performs well on electroencephalogram signals data. Task-1 helped in understanding the relation between the features with the help of plots. In the task-2, performed multiple statistical calculations to filter the better performing models followed by selecting one. Though there are only three parameters for model 2, it performed the least in all calculations. Moreover, model 5 and 3 performed well in all the calculations, coming to AIC and BIC values, model 3 has shown better results for this data. This selected model-3 has shown best results on train data and also, on evaluations.

Reference

Kenton, W. (2020) What Is a Time Series. [online] Available at: https://www.investopedia.com/terms/t/timeseries.asp (Accessed: 01 February 2021)

Barone, A. (2021) Residual Sum of Squares (RSS). [online] Available at: https://www.investopedia.com/terms/r/residual-sum-of-squares.asp#:~:text=A%20residual%20sum%20of%20squares,the%20residuals%2C%20or%20error%20term (Accessed: 01 February 2021)

Wikipedia (2021) Akaike information criterion. [online] Available at: https://en.wikipedia.org/wiki/Akaike_information_criterion (Accessed: 02 February 2021)

Bevans, R. (2020) An introduction to the Akaike information criterion. [online] Available at: https://www.scribbr.com/statistics/akaike-information-criterion/ (Accessed: 04 February 2021)

Burn, R. (2020) What to Do When Your Model Has a Non-Normal Error Distribution. [online] Available at: https://towardsdatascience.com/what-to-do-when-your-model-has-a-non-normal-error-distribution-f7c3862e475f#:~:text=An%20error%20distribution%20is%20a,important%20than%20the%20point%20prediction.&text=A%20point%20prediction%20tells%20us,are%20likely%20to%20be%20distributed (Accessed: 04 February 2021)

Michael. (2020) Meaning of error bars in JASP's QQ plot? [online]. Available at: https://forum.cogsci.nl/discussion/6392/meaning-of-error-bars-in-jasps-qq-plot
(Available at: 06 February 2021)

Appendix

Code chunk – (1)

Read csv data

```
X = read.csv("X.csv", header = F)
y = read.csv("y.csv", header = F)
time = read.csv("time.csv", header = F, skip = 1)
#fix the time values
time = rbind(0, time)

#generating matrix forms
X_m <- data.matrix(X)
y_m <- data.matrix(y)

#generating data frame
colnames(X) = paste0(rep("x",ncol(X)),1:ncol(X))
colnames(y) = "y"
colnames(time) = "time"

df = cbind(time,X,y)
head(df,4)
```

Task 1 Preliminary data analysis

Code chunk – (2)

a) Time Series plots

```
library(ggplot2)
library(reshape2)

melt_df <- melt(df,id="time")

par(mfrow=c(2,2))

plot(df$time,df$x1, type='l', col="sienna2", xlab="Time (sec)", ylab="x1")

plot(df$time,df$x2, type='l', col="palegreen3", xlab="Time (sec)", ylab="x2")
plot(df$time,df$x3, type='l', col="yellow3", xlab="Time (sec)", ylab="x3")
plot(df$time,df$x4, type='l', col="black", xlab="Time (sec)", ylab="x4")
mtext("Time series plots", outer=TRUE, cex=1, line=-3.5)

plot(df$time,df$y, type='l', col="darkblue", xlab="Time (sec)", ylab="y")

ggplot(melt_df,aes(x=time,y=value,colour=variable,group=variable)) + geom_line() + ggtitle("Time Series plot") + theme(pl
ot.title = element_text(hjust = 0.5)) + xlab("Time (sec)") + ylab("Signals")
```

Code chunk – (3)

b) Distribution for each EEG signal

```
library(Hmisc)

hist.data.frame(df[ , 2:6])
mtext("Distribution for each signal", outer=TRUE, cex=1, line=-2.5)
```

Code chunk – (4)

c) Correlation and scatter plots

```
library(corrplot)
group <- NA
group[cor(df[,2:6]) < - 0.5] <- 1
group[cor(df[,2:6]) >= - 0.5 & cor(df[,2:6]) <= 0.5] <- 2
group[cor(df[,2:6]) > 0.5] <- 3

cor(df[,2:6])
```

```
pairs(df[,2:6], col=c('black', 'plum3', 'sienna2')[group])
mtext("Scatter plots", outer=TRUE, cex=1, line=-2)
```

Task 2 Regression - modelling the relationship between EEG signals

Code chunk – (5)

```
thetaBias = matrix(1 , nrow=length(X_m[,1]),ncol=1)

#Generate candidate models
X1<-cbind(X_m[,4],X_m[,1]^2,X_m[,1]^3,X_m[,3]^4,thetaBias)
X2<-cbind(X_m[,3]^3,X_m[,3]^4,thetaBias)
X3<-cbind(X_m[,2],X_m[,1]^3,X_m[,3]^4,thetaBias)
X4<-cbind(X_m[,4],X_m[,1]^3,X_m[,3]^4,thetaBias)
X5<-cbind(X_m[,4],X_m[,1]^2,X_m[,1]^3,X_m[,3]^4,X_m[,1]^4,thetaBias)
```

Code chunk – (6)

2.1 Estimate model parameters

```
to_thetaHat <- function(x_th,y_th) {
  thetaa <- solve(t(x_th) %*% x_th) %*% t(x_th) %*% y_th
  return(thetaa)
}

thetaHat1<- to_thetaHat(X1,y_m)
thetaHat2<- to_thetaHat(X2,y_m)
thetaHat3<- to_thetaHat(X3,y_m)
thetaHat4<- to_thetaHat(X4,y_m)
thetaHat5<- to_thetaHat(X5,y_m)
```

Code chunk – (7)

2.2 Model residual (error) sum of squared errors (RSS)

```
to_yHat <- function(x_rss,thetahat_rss) {
  y_hat = x_rss %*% thetahat_rss
  return(y_hat)
}

to_RSS <- function(x_rss, y_rss, thetahat_rss) {
  y_hat = to_yHat(x_rss, thetahat_rss)
  rss_val <- sum((y_rss-y_hat)^2)
  return(rss_val)
}

RSS1<- to_RSS(X1,y_m, thetaHat1)
RSS2<- to_RSS(X2,y_m, thetaHat2)
RSS3<- to_RSS(X3,y_m, thetaHat3)
RSS4<- to_RSS(X4,y_m, thetaHat4)
RSS5<- to_RSS(X5,y_m, thetaHat5)
```

Code chunk – (8)

2.3 log-likelihood function

```
to_loglikelihood <- function(x_rss,rss_li) {
  n<-length(x_rss[,1])
  variance<- rss_li/(n-1)
  likelihood<- -((n/2)*log(2*pi))-((n/2)*log(variance))-((1/(2*variance))*rss_li)
  return(likelihood)
}

likelihood1<-to_loglikelihood(X1,RSS1)
likelihood2<-to_loglikelihood(X2,RSS2)
likelihood3<-to_loglikelihood(X3,RSS3)
likelihood4<-to_loglikelihood(X4,RSS4)
likelihood5<-to_loglikelihood(X5,RSS5)
```

Code chunk – (9)

2.4 AIC and BIC

```
n<-length(X[,1])

to_AicBic <- function(thetaHat_ic,likelihood_ic) {
  k<-length(thetaHat_ic)
  aic <- (2*k)-(2*likelihood_ic)
  bic <- (k*log(n))-(2*likelihood_ic)
  return(list(aic,bic))
}

aicbic1<-matrix(to_AicBic(thetaHat1,likelihood1),1,2)
aicbic2<-matrix(to_AicBic(thetaHat2,likelihood2),1,2)
aicbic3<-matrix(to_AicBic(thetaHat3,likelihood3),1,2)
aicbic4<-matrix(to_AicBic(thetaHat4,likelihood4),1,2)
aicbic5<-matrix(to_AicBic(thetaHat5,likelihood5),1,2)
```

Code chunk – (10)

2.5 Plot Error distributions

```
to_error <- function(y_e, yHat_e) {
  e <- y_e - yHat_e
  return(e)
}

yHat1 <- to_yHat(X1,thetaHat1)
yHat2 <- to_yHat(X2,thetaHat2)
yHat3 <- to_yHat(X3,thetaHat3)
yHat4 <- to_yHat(X4,thetaHat4)
yHat5 <- to_yHat(X5,thetaHat5)

par(mfrow=c(2,3))

qqplot(X_m, y_m, xlab = "input signals X", ylab = "output signal y", main="Model 1")
qqline(yHat1)
qqplot(X_m, y_m, xlab = "input signals X", ylab = "output signal y", main="Model 2")
qqline(yHat2)
qqplot(X_m, y_m, xlab = "input signals X", ylab = "output signal y", main="Model 3")
qqline(yHat3)
qqplot(X_m, y_m, xlab = "input signals X", ylab = "output signal y", main="Model 4")
qqline(yHat4)
qqplot(X_m, y_m, xlab = "input signals X", ylab = "output signal y", main="Model 5")
qqline(yHat5)

mtext("Error distribution for each model", outer=TRUE, cex=1, line=-1.3)

qqplot(X_m,y_m,xlab = "input signals X", ylab = "output signal y")
qqline(yHat1, col='sienna2')
qqline(yHat2, col='palegreen3')
qqline(yHat3, col='yellow3')
qqline(yHat4, col='black')
qqline(yHat5, col='darkblue')
legend("topleft", legend = c("Model 1","Model 2","Model 3","Model 4","Model 5"), col=c("sienna2", "palegreen3",  "yello
w3", "black", "darkblue"),pch=c(4,4,4,4,4))

mtext("Error distribution of all models", outer=TRUE, cex=1, line=-3.3)
```

Code chunk – (11)

2.7 Model validation

```
#train test split
X_train <- head(X_m, round(nrow(X_m)*0.7))
X_test <- tail(X_m, round(nrow(X_m)*0.3))
```

```
y_train <- head(y_m, round(nrow(y_m)*0.7))
y_test <- tail(y_m, round(nrow(y_m)*0.3))

#Best Model
to_model3 <- function(dataa) {
  thetaBias_d = matrix(1 , nrow=length(dataa[,1]),ncol=1)
  X3_mod <-cbind(dataa[,2],dataa[,1]^3, dataa[,3]^4,thetaBias_d)
  return(X3_mod)
}



# 2.7 1. Estimate model parameters on train data
thetaHat3_test <- to_thetaHat(to_model3(X_train), y_train)

# 2.7 2. Model's prediction on test data

yHat3_test <- to_yHat(to_model3(X_test), thetaHat3_test)

# 2.7 3. 95% Confidence intervals and plot error bars

error = y_test - yHat3_test



sse = norm(error , type = "2")^2



sigma_2 = sse/(length(to_model3(X_test)[,1]) - 1 )

cov_thetaHat =  solve(t(to_model3(X_test)) %*% to_model3(X_test))

cov_thetaHat

var_yHat = matrix(0 , length(to_model3(X_test)[,1]))

number_of_parameters<-length(thetaHat3_test)

number_of_parameters

for( i in 1:length(to_model3(X_test)[,1])){

  X_test_i = matrix( to_model3(X_test)[i,] , 1 , number_of_parameters )

  var_yHat[i,1] = sigma_2 * (X_test_i %*% cov_thetaHat %*% t(X_test_i) )

}



CI = 1.96 * sqrt(var_yHat) # Confidence interval

qqplot(X_test, yHat3_test , type = "l", xlab = "X test input signals", ylab = "Y test output signal with Confidence intervals")

segments(X_test, yHat3_test-CI, X_test, yHat3_test+CI)

mtext("Error bars", outer=TRUE, cex=1.5, line=-3.3)
```

Task 3

Code chunk – (12)

3.1 Two largest value parameters

```
thetaHat3[4]
```

```
thetaHat3[1]
```

Code chunk – (13)

3.2 Uniform distribution

```
sample_theta1 <- runif(1e4,(thetaHat3[1]-(0.5*thetaHat3[1])), (thetaHat3[1]+(0.5*thetaHat3[1])))
length(sample_theta1)

sample_theta4 <- runif(1e4,(thetaHat3[4]-(0.5*thetaHat3[4])), (thetaHat3_test[4]+(0.5*thetaHat3_test[4])))
sample_theta4[3]
```

Code chunk – (14)

3.3 Rejection ABC method

```
mse_main <- sum((y_m-yHat3)^2)/201

sample_theta1_accept = list()
sample_theta4_accept = list()

for (i in 1:1e4) {
  sample_thetahat <- matrix(c(sample_theta1[i], thetaHat3[2], thetaHat3[3], sample_theta4[i]), nrow=4, ncol=1)
  sample_yHat <- X3 %*% sample_thetahat
  sample_mse <- (1/length(X_m[,1]))*sum((y_m-sample_yHat)^2)
  #print(sample_mse)
  if (sample_mse <= 0.308) {
    sample_theta1_accept <- append(sample_theta1_accept,sample_theta1[i])
    sample_theta4_accept <- append(sample_theta4_accept,sample_theta4[i])
  }
}

length(sample_theta1_accept)

max(sapply(sample_theta1_accept, max))

length(sample_theta4_accept)

max(sapply(sample_theta4_accept, max))
```

Code chunk – (15)

3.4 Joint and marginal posterior distribution plots

```
library(tidyverse)

library(magrittr)

matrix_parameters <- cbind(as.matrix(as.numeric(sample_theta1_accept)), as.matrix(as.numeric(sample_theta4_accept)))

df_parameters <- as.data.frame(matrix_parameters)

#dev.off()
```

Code chunk – (16)

```
### Joint plot
ggplot(df_parameters, aes(x=df_parameters[,1], y=df_parameters[,2]) ) + stat_density_2d(aes(fill = ..level..), geom = "polyg
on", colour="white") + xlab("sample parameters of theta1") + ylab("sample parameters of theta4") + ggtitle("Joint posterior
distribution")

ggplot(df_parameters, aes(x=df_parameters[,1], y=df_parameters[,2]))+ theme_bw() + geom_density2d(color=' black') + ge
om_point(alpha=0.5, col='red') + ggtitle("Sample parameters") + xlab("Parameter 1") + ylab("Parameter 4")
```

Code chunk – (17)

```
### Marginal plot
par(mfrow=c(1,2))
hist(as.numeric(sample_theta1_accept) , main="Parameter 1", xlab='Samples of Parameter 1')
hist(as.numeric(sample_theta4_accept), main="Parameter 4", xlab ='Samples of Parameter 4')
mtext("Marginal posterior distributions for two parameters", outer=TRUE, cex=1.5, line=-1.2)
```