MACHINE LEARNING PROJECT

Eric Brown

Abstract: Exploring two types of datasets (transactions of online based company & wine data) and performing data cleaning tasks to prepare datasets for analysis. Once data collection and data cleaning process are complete, machine learning algorithms are used to conduct time series analysis on the prescribed datasets. In order to answer questions like which model best predicts the outcome variable of the wine dataset. As well as, which model best predicts the future purchase amount from the online retail store website.

Table of Contents

Problem 1 – Retail data modeling: identifying the insights:

Source: https://archive.ics.uci.edu/ml/datasets/Online+Retail

Data Set Information:

This is a transnational data set which contains all the transactions occurring between 12/01/2010 and 12/09/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

Attribute Information:

InvoiceNo: Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.

StockCode: Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.

Description: Product (item) name. Nominal.

Quantity: The quantities of each product (item) per transaction. Numeric.

InvoiceDate: Invice Date and time. Numeric, the day and time when each transaction was generated.

UnitPrice: Unit price. Numeric, Product price per unit in sterling.

CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.

Country: Country name. Nominal, the name of the country where each customer resides.

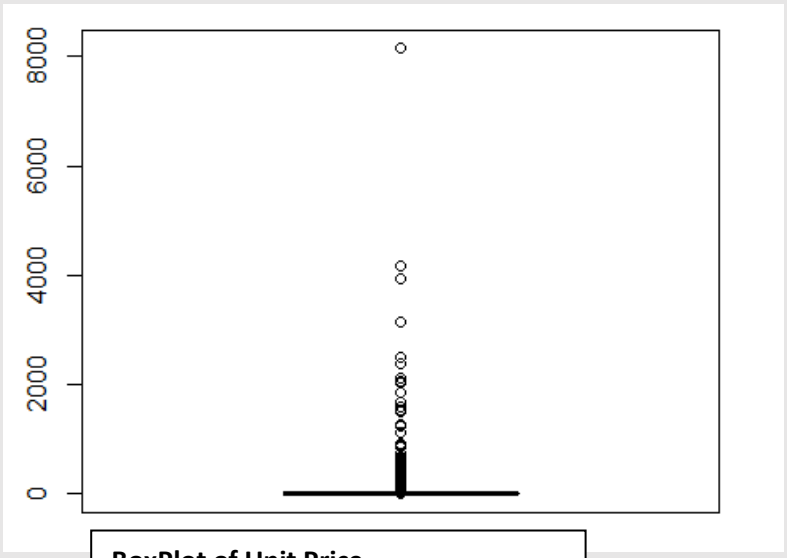
Task 1: Data Exploration

Show how you can explore the data (e.g., structure, aggregation, missing data, outliers, and transformation if needed) and formulate a time series problem.

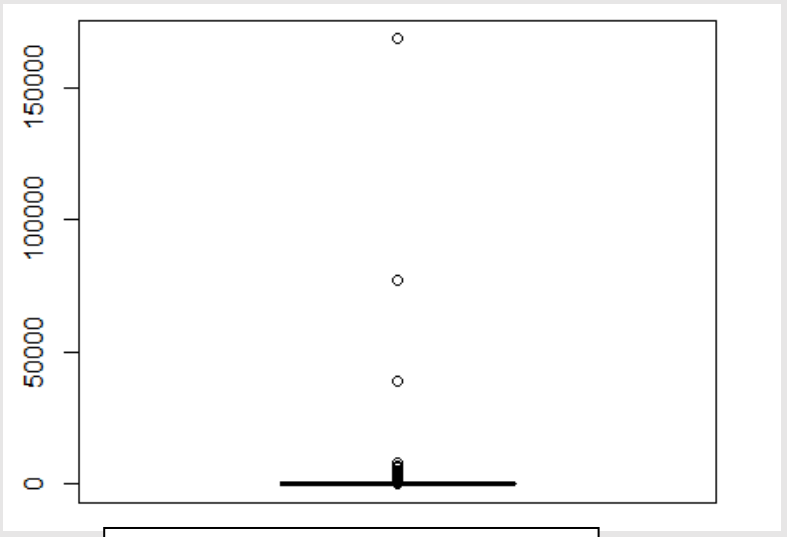
```
> #Final exam
> #Dataset 1
> #This is a transnational dataset which contains all the transactions occurring between 12/01/2010
> #and 12/09/2011 for a UK-based and registered non-store online retail
> library(readxl)
> transData<-read_excel("C:/Users/esbro/Desktop/IE 575/Final_Exam/OnlineRetail.xlsx") #importing dataset for question 1
> #exploring summary statistics of the dataset
> summary(transData)
  InvoiceNo          StockCode         Description          Quantity          InvoiceDate
 Length:541909      Length:541909      Length:541909      Min.   :-80995.00   Min.   :2010-12-01 08:26:00
 Class :character   Class :character   Class :character   1st Qu.:     1.00   1st Qu.:2011-03-28 11:34:00
 Mode  :character   Mode  :character   Mode  :character   Median :     3.00   Median :2011-07-19 17:17:00
                                                          Mean   :     9.55   Mean   :2011-07-04 13:34:57
                                                          3rd Qu.:    10.00   3rd Qu.:2011-10-19 11:27:00
                                                          Max.   : 80995.00   Max.   :2011-12-09 12:50:00
                                                          NA's   :4
   UnitPrice           CustomerID       Country
 Min.   :-11062.06   Min.   :12346    Length:541909
 1st Qu.:     1.25   1st Qu.:13953    Class :character
 Median :     2.08   Median :15152    Mode  :character
 Mean   :     4.61   Mean   :15288
 3rd Qu.:     4.13   3rd Qu.:16791
 Max.   : 38970.00   Max.   :18287
                     NA's   :135080
> #handling missing data issues
> colSums(is.na(transData))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0        1454           4           0           0      135080           0
> names(which(colSums(is.na(transData))>0))
[1] "Description" "Quantity"    "CustomerID"
> transData2<-na.omit(transData)
> colSums(is.na(transData2))
  InvoiceNo   StockCode Description    Quantity InvoiceDate   UnitPrice  CustomerID     Country
          0           0           0           0           0           0           0           0
> transData2<-subset(transData2,transData2$Quantity>=0)
> #checking to make sure there are no missing values in any variables anymore in the dataset
> summary(transData2$Quantity)
    Min.  1st Qu.    Median     Mean  3rd Qu.     Max.
    1.00     2.00     6.00    13.02    12.00 80995.00
>
```

```
> #checking to see if their are any outlier data points in the dataset
> summary(transData2)
  InvoiceNo          StockCode         Description          Quantity         InvoiceDate
 Length:397921      Length:397921      Length:397921      Min.   :    1.00   Min.   :2010-12-01 08:26:00
 Class :character   Class :character   Class :character   1st Qu.:    2.00   1st Qu.:2011-04-07 11:12:00
 Mode  :character   Mode  :character   Mode  :character   Median :    6.00   Median :2011-07-31 14:39:00
                                                          Mean   :   13.02   Mean   :2011-07-10 23:44:18
                                                          3rd Qu.:   12.00   3rd Qu.:2011-10-20 14:33:00
                                                          Max.   :80995.00   Max.   :2011-12-09 12:50:00
   UnitPrice           CustomerID      Country           purchaseAm
 Min.   :    0.000   Min.   :12346   Length:397921      Min.   :     0.00
 1st Qu.:    1.250   1st Qu.:13969   Class :character   1st Qu.:     4.68
 Median :    1.950   Median :15159   Mode  :character   Median :    11.80
 Mean   :    3.116   Mean   :15294                      Mean   :    22.39
 3rd Qu.:    3.750   3rd Qu.:16795                      3rd Qu.:    19.80
 Max.   :8142.750    Max.   :18287                      Max.   :168469.60
> boxplot(transData2$UnitPrice)
> boxplot(transData2$purchaseAm)
> #trying to get rid of the outlier data points by
> #removing rows from the data frame that have a value in the listed columns that are 1.5 times the interquartile range
> #greater than the third quartile (Q3) or 1.5 times the interquartile range less than the first quartile (Q1)
> Q1<- quantile(transData2$UnitPrice, .25)
> Q3<- quantile(transData2$UnitPrice, .75)
> IQR<- IQR(transData2$UnitPrice)
> transData3<- subset(transData2, transData2$UnitPrice> (Q1 - 1.5*IQR) & transData2$UnitPrice< (Q3 + 1.5*IQR))
> Q1_2<- quantile(transData2$Quantity, .25)
> Q3_2<- quantile(transData2$Quantity, .75)
> IQR_2<- IQR(transData2$Quantity)
> transData3<- subset(transData3, transData3$Quantity> (Q1_2 - 1.5*IQR) & transData3$Quantity< (Q3_2 + 1.5*IQR))
>
```

```
> #exploring dataset to see which customers had most amount of orders from the UK-based company
> #grouping data by customers to count invoice orders per customer
> customerOrders <-
+   aggregate(x = transData2$InvoiceNo,            # Specify data column
+             by = list(transData2$CustomerID),    # Specify group indicator
+             FUN = function(x) length(unique(x))) #Desired function
> colnames(customerOrders)<-c('CustomerID','Number_Orders')
> #outputting top customers who ordered the most from the company
> head(customerOrders[order(customerOrders$Number_Orders,decreasing=TRUE),])
     CustomerID Number_Orders
327       12748           210
1881      14911           201
4012      17841           124
563       13089            97
1663      14606            93
2178      15311            91
> #adding the purchased amount of each invoice/order to the original dataset
> transData2$purchaseAm<-transData2$UnitPrice*transData2$Quantity
> #exploring dataset to see which customer purchased the most from the UK-based company
> customerAmount<-
+   aggregate(x = transData2$purchaseAm,          # Specify data column
+             by = list(transData2$CustomerID),   # Specify group indicator
+             FUN = sum )                         #Desired function
> colnames(customerAmount)<-c('CustomerID','Order_Amount')
> #outputting top customers who purchased the most from the company
> head(customerAmount[order(customerAmount$Order_Amount,decreasing=TRUE),])
     CustomerID Order_Amount
1691      14646     280206.0
4203      18102     259657.3
3730      17450     194550.8
3010      16446     168472.5
1881      14911     143825.1
56        12415     124914.5
```

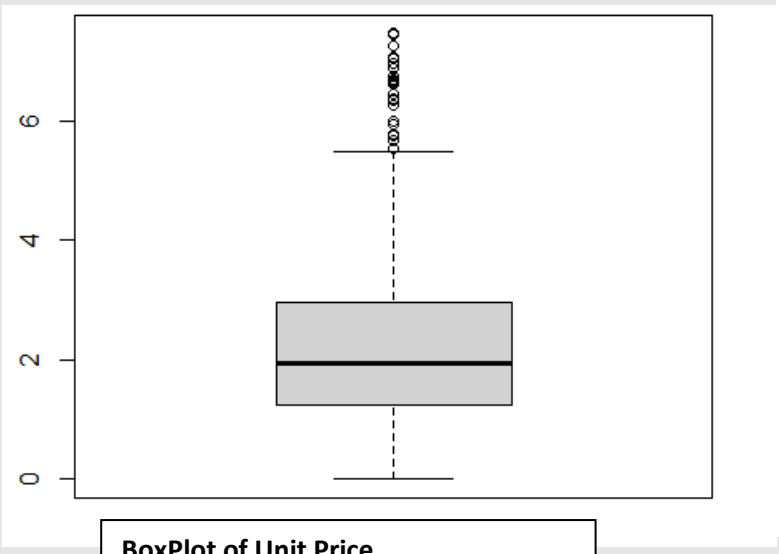


BoxPlot of Unit Price

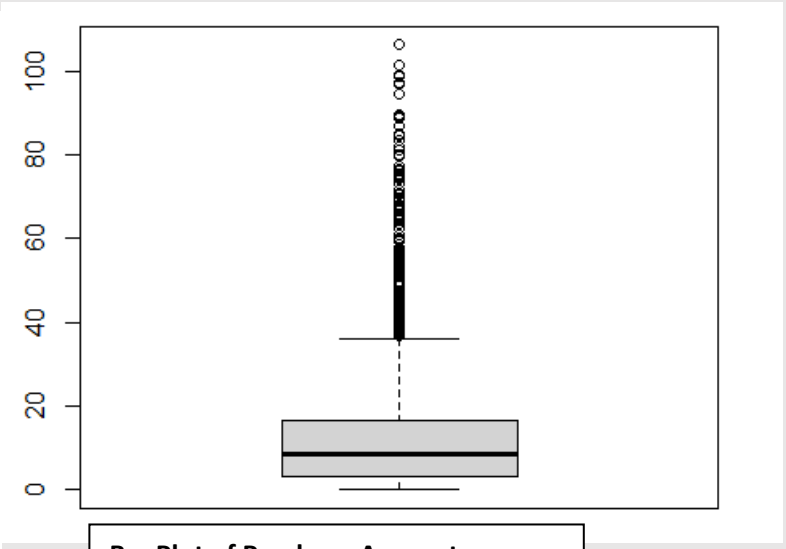


BoxPlot of Purchase Amount

```
> #checking to see if the prices per unit and invoice dollar amount are more evenly distributed
> boxplot(transData3$UnitPrice)
> boxplot(transData3$purchaseAm)
> summary(transData3)
  InvoiceNo          StockCode          Description           Quantity       InvoiceDate                    UnitPrice
 Length:298750      Length:298750      Length:298750      Min.    : 1.000   Min.    :2010-12-01 08:26:00   Min.    :0.000
 Class :character   Class :character   Class :character   1st Qu.: 2.000   1st Qu.:2011-04-07 12:38:00   1st Qu.:1.250
 Mode  :character   Mode  :character   Mode  :character   Median : 4.000   Median :2011-08-03 15:28:00   Median :1.950
                                                          Mean    : 5.515   Mean    :2011-07-12 19:57:59   Mean    :2.312
                                                          3rd Qu.:10.000   3rd Qu.:2011-10-24 13:34:00   3rd Qu.:2.950
                                                          Max.    :15.000   Max.    :2011-12-09 12:50:00   Max.    :7.460

   CustomerID      Country            purchaseAm
 Min.    :12347   Length:298750      Min.    :  0.00
 1st Qu.:14056   Class :character   1st Qu.:  3.32
 Median :15311   Mode  :character   Median :  8.50
 Mean    :15368                      Mean    : 11.05
 3rd Qu.:16891                      3rd Qu.: 16.50
 Max.    :18287                     Max.    :106.20
```



BoxPlot of Unit Price

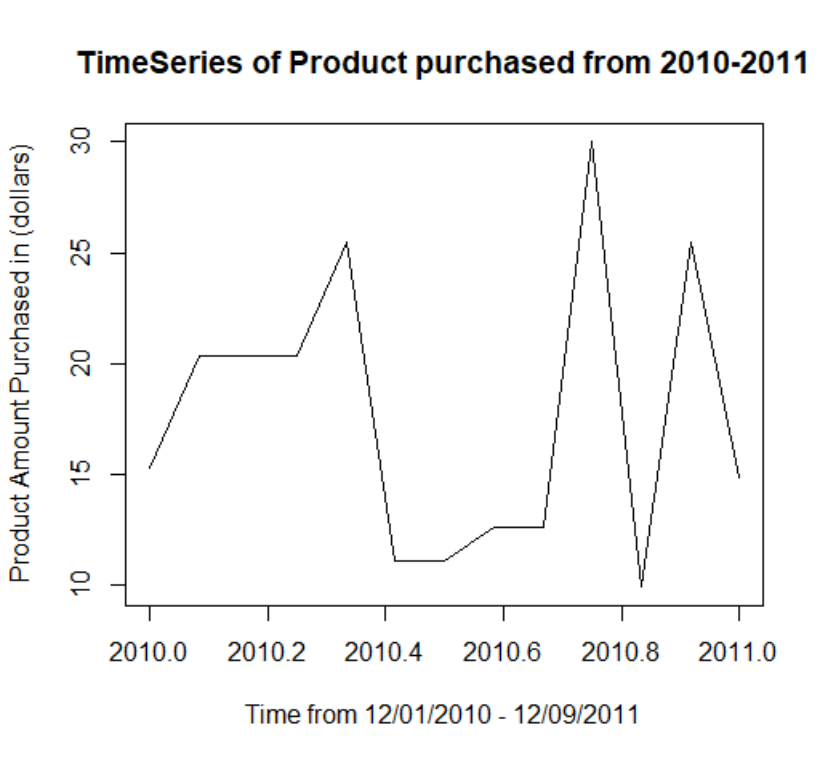


BoxPlot of Purchase Amount

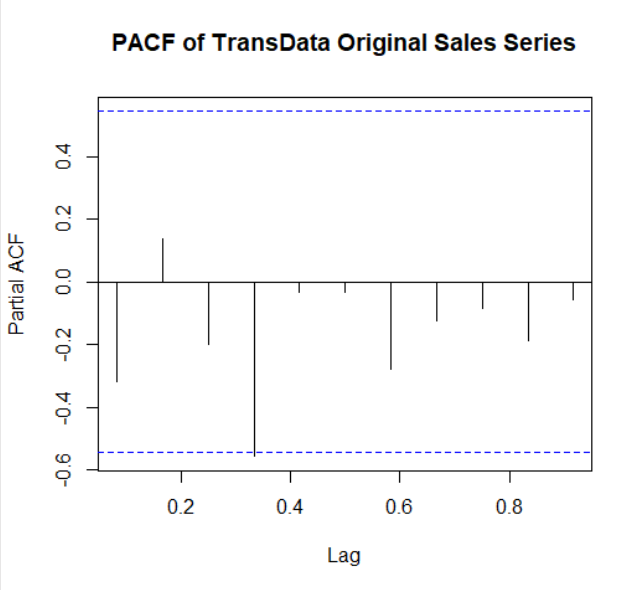
Task 2: Predictive Modeling

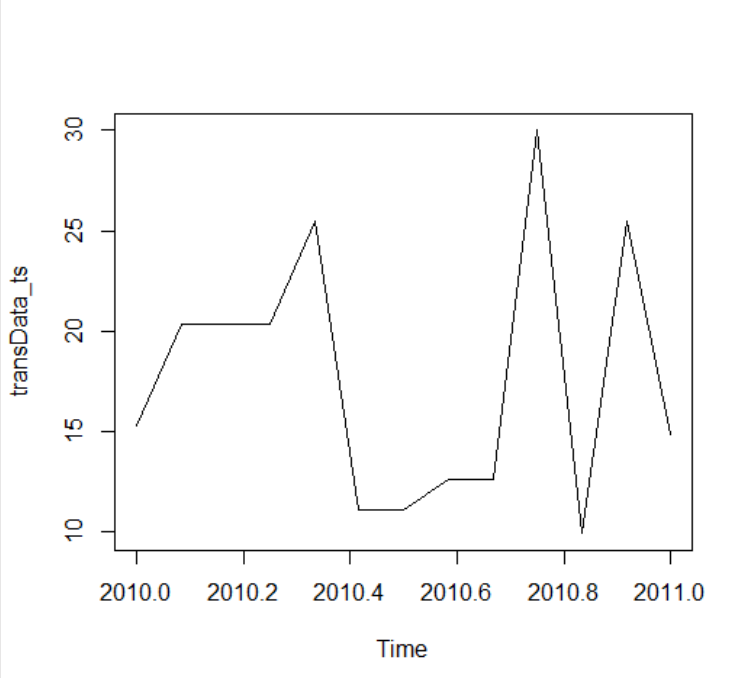
Build at least two ARIMA models for your formulated problem.

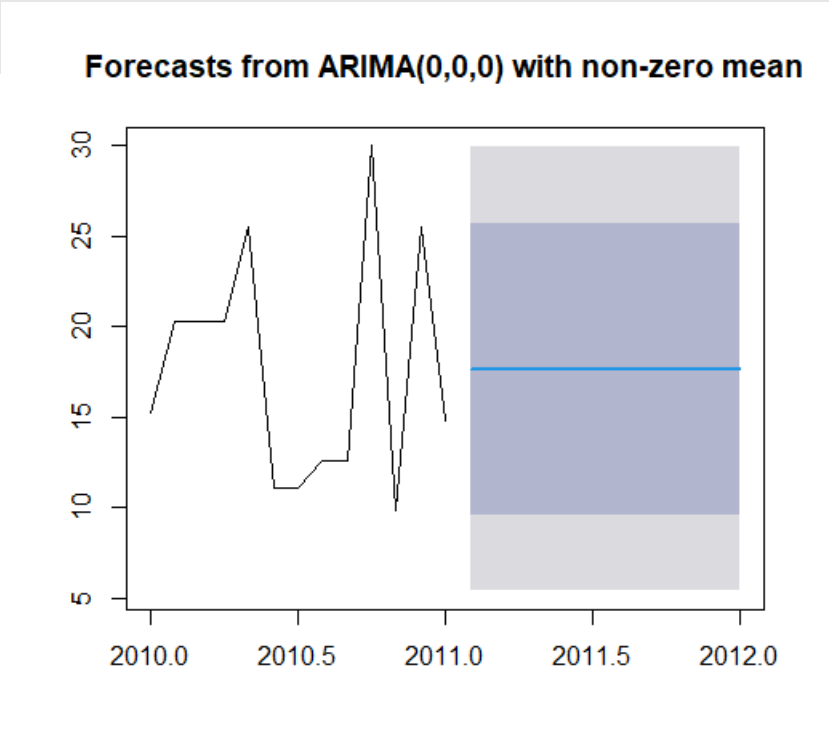
```
> #formulating time series of data
> transData_ts<-ts(transData3$purchaseAm,start=c(2010),end=c(2011),frequency = 12)
> plot.ts(transData_ts,xlab="Time from 12/01/2010 - 12/09/2011",ylab="Product Amount Purchased in (dollars)",
+         main="TimeSeries of Product purchased from 2010-2011")
> #checking to see if the time series is stationary
> library(tseries)
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

    'tseries' version: 0.10-51

    'tseries' is a package for time series analysis and computational finance.

    See 'library(help="tseries")' for details.

Warning message:
package 'tseries' was built under R version 4.1.3
> adf.test(transData_ts, alternative ="stationary")

        Augmented Dickey-Fuller Test

data:  transData_ts
Dickey-Fuller = -1.5839, Lag order = 2, p-value = 0.7309
alternative hypothesis: stationary

> #using PACF to identify the number of lags for AR model and spikes in ACF to determine
> #the number of error lags of MA models
> acf(transData_ts, main="ACF of TransData Original Sales Series")
> pacf(transData_ts,main="PACF of TransData Original Sales Series")
```




Trained Sales Model
2010.0
2010.2
2010.4
2010.6
2010.8
2011.0
ACF
PACF
Lag
Lag


PACF of TransData Original Sales Series
Partial ACF
Lag

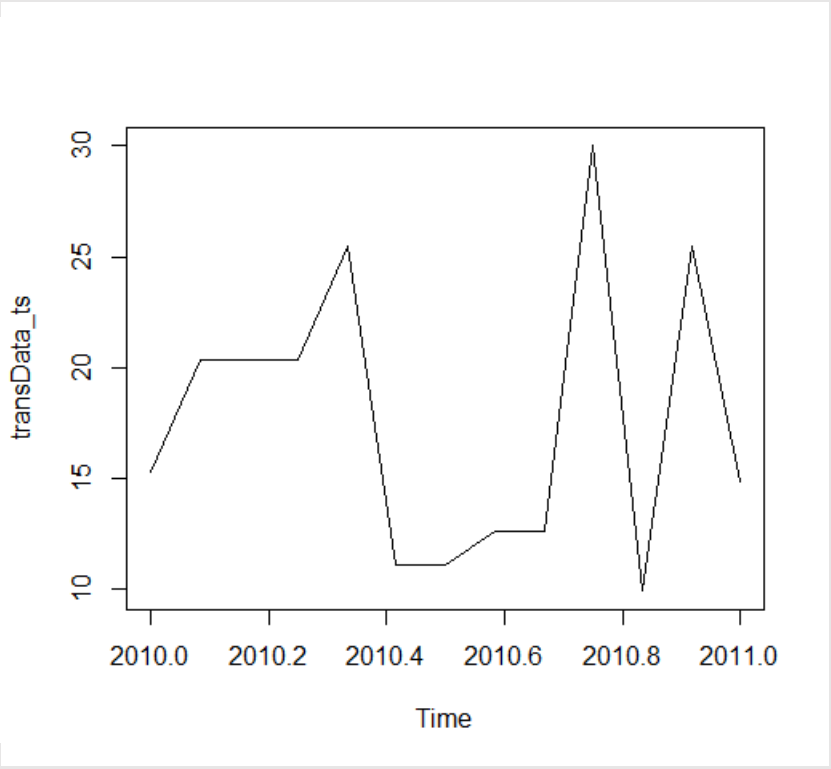

transData_ts
Time
2010.0
2010.2
2010.4
2010.6
2010.8
2011.0

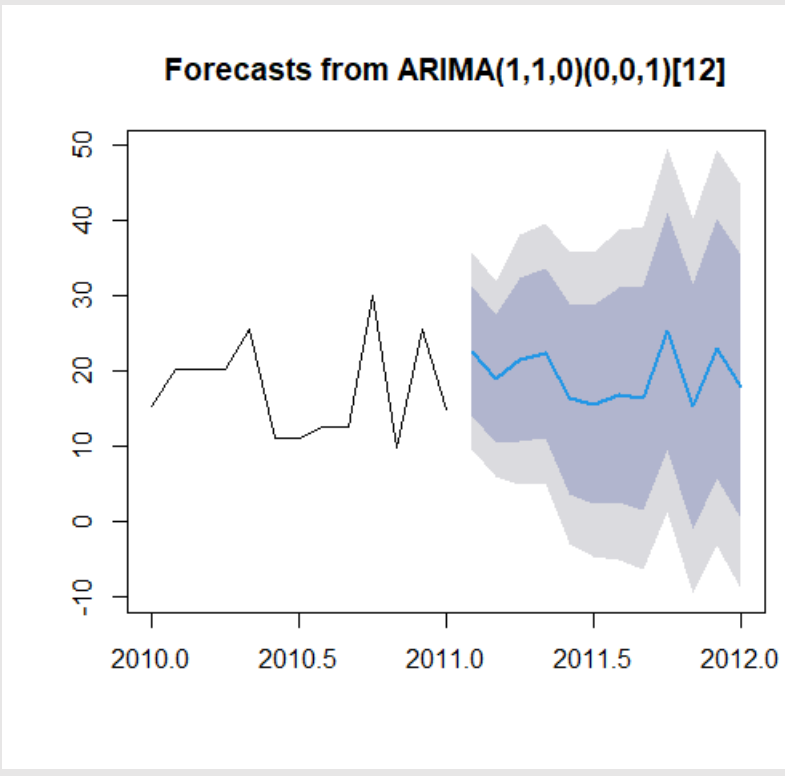

Forecasts from ARIMA(0,0,0) with non-zero mean
2010.0
2010.5
2011.0
2011.5
2012.0

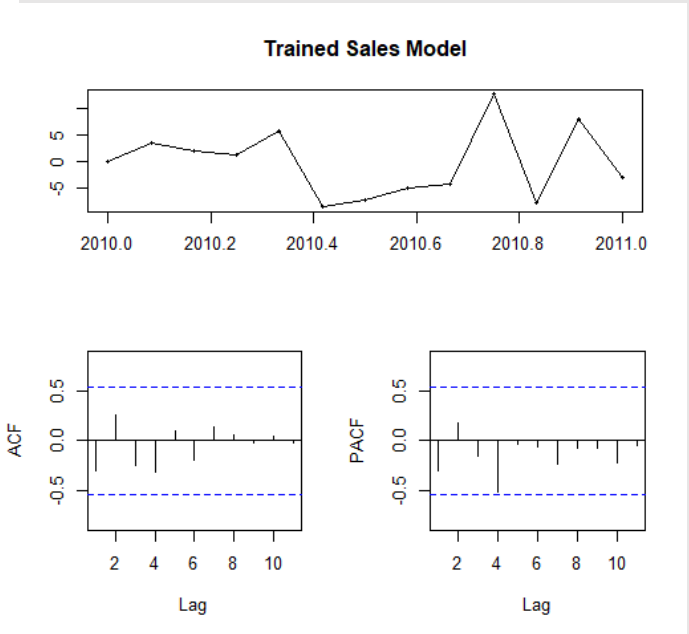
```
> #ARIMA model 2
> opt_sales2 <- arima(transData_ts, order=c(1,1,0),seasonal = list(order = c(0,0,1), period=12))
> opt_sales2$aic
[1] 88.12951
> tsdisplay(residuals(opt_sales2), lag.max=11,main ='Trained Sales Model')
> #predicted values for arima model 2
> opt_sales_forecast2 <- forecast(opt_sales2, h=12)
> opt_sales_forecast2
         Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
Feb 2011       22.64877 14.0621948 31.23535  9.516735 35.78081
Mar 2011       18.98385 10.5305696 27.43714  6.055671 31.91204
Apr 2011       21.52560 10.7241718 32.32703  5.006240 38.04496
May 2011       22.34668 11.1362635 33.55710  5.201827 39.49154
Jun 2011       16.35829  3.7409179 28.97567 -2.938317 35.65490
Jul 2011       15.53187  2.3389796 28.72476 -4.644915 35.70866
Aug 2011       16.83381  2.5871188 31.08050 -4.954624 38.62224
Sep 2011       16.48244  1.6536917 31.31119 -6.196175 39.16106
Oct 2011       25.38020  9.6346025 41.12581  1.299384 49.46103
Nov 2011       15.25950 -0.9833035 31.50230 -9.581723 40.10072
Dec 2011       23.01266  5.8702332 40.15508 -3.204419 49.22974
Jan 2012       17.85830  0.4040077 35.31260 -8.835738 44.55235
> #plotting predicted values
> plot.ts(transData_ts)
> plot(opt_sales_forecast2)
```

```
> #Building four ARIMA models for the formulated time series created above of the transnational data
> library(forecast)
Warning message:
package 'forecast' was built under R version 4.1.3
> #retrieving optimal parameters for the ARIMA model
> auto.arima(transData_ts,seasonal=TRUE)
Series: transData_ts
ARIMA(0,0,0) with non-zero mean

Coefficients:
        mean
      17.6515
s.e.   1.7333

sigma^2 = 42.31:  log likelihood = -42.27
AIC=88.54   AICc=89.74   BIC=89.67
> #ARIMA model 1 - with auto.arima prescribed parameters
> opt_sales <- arima(transData_ts, order=c(0,0,0))
> opt_sales$aic #outputting aic score to compare arima models on
[1] 88.53692
> tsdisplay(residuals(opt_sales), lag.max=11,main ='Trained Sales Model')
> #predicted values for arima model 1
> opt_sales_forecast <- forecast(opt_sales, h=12)
> opt_sales_forecast
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Feb 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Mar 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Apr 2011       17.65154 9.642644 25.66043 5.402991 29.90009
May 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Jun 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Jul 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Aug 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Sep 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Oct 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Nov 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Dec 2011       17.65154 9.642644 25.66043 5.402991 29.90009
Jan 2012       17.65154 9.642644 25.66043 5.402991 29.90009
> #plotting predicted values
> plot.ts(transData_ts)
> plot(opt_sales_forecast)
```

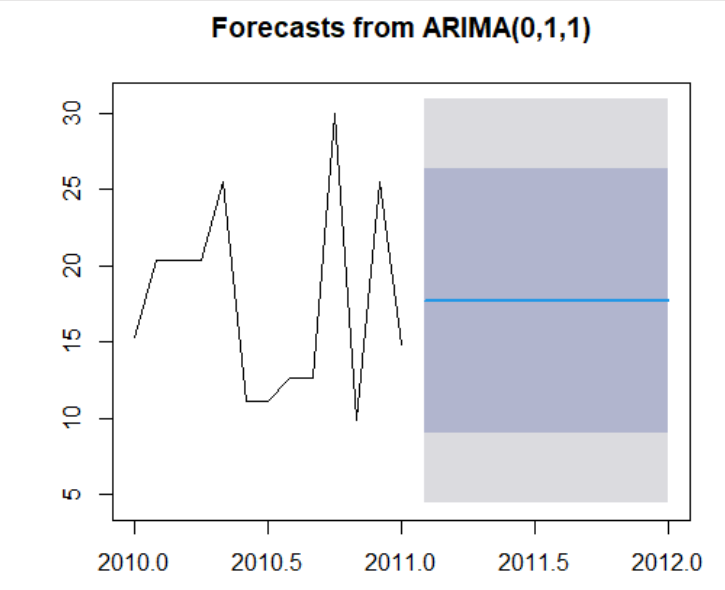

Trained Sales Model
2010.0
2010.2
2010.4
2010.6
2010.8
2011.0
ACF
PACF
Lag
Lag

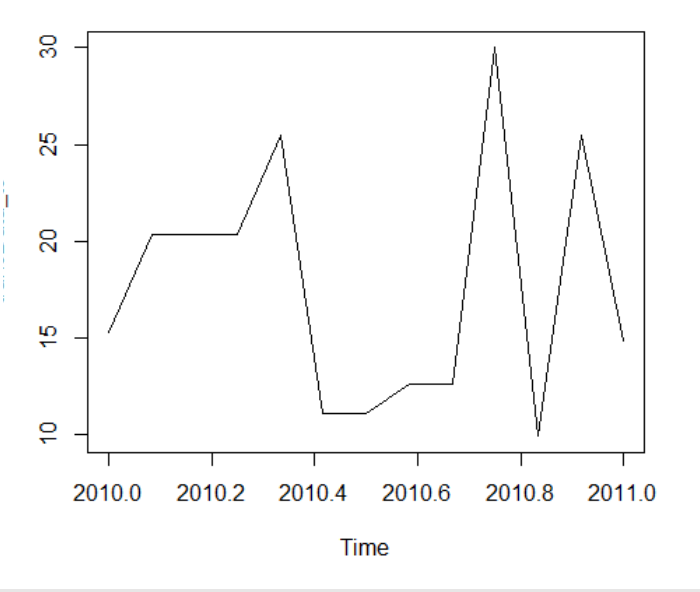

transData_ts
Time
2010.0
2010.2
2010.4
2010.6
2010.8
2011.0


Forecasts from ARIMA(1,1,0)(0,0,1)[12]
2010.0
2010.5
2011.0
2011.5
2012.0

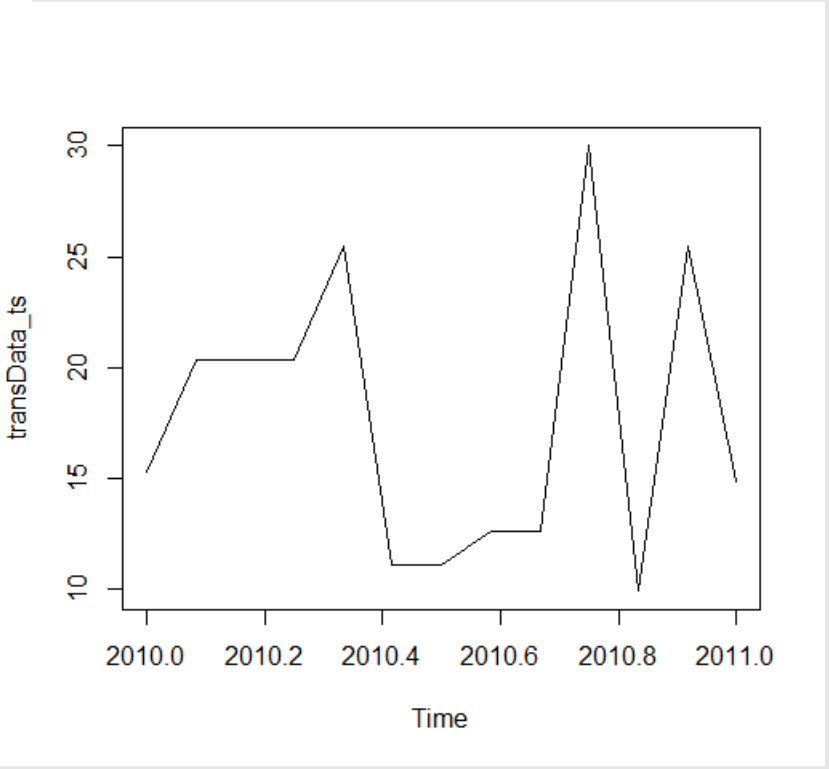

Trained Sales Model
2010.0
2010.2
2010.4
2010.6
2010.8
2011.0
ACF
PACF
Lag
Lag

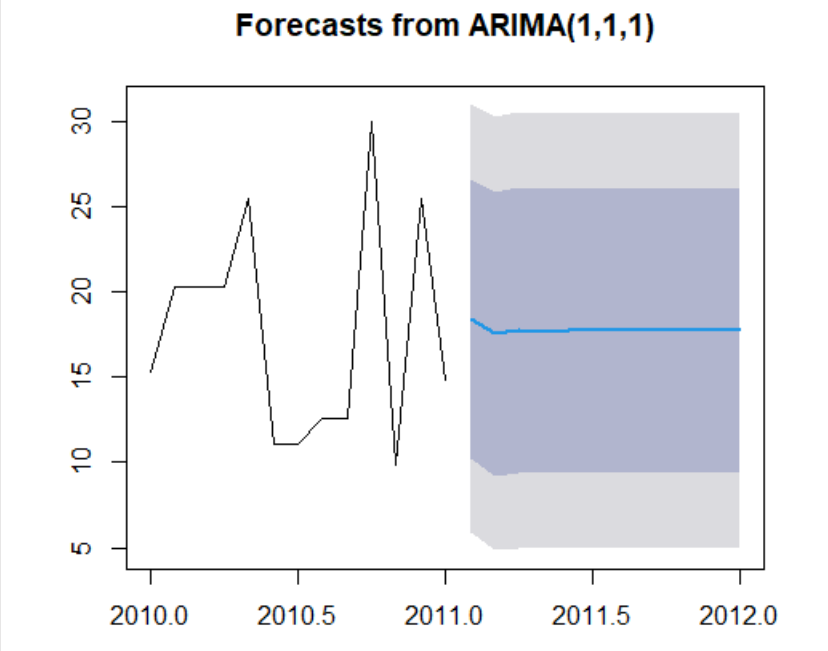
```
> #ARIMA model 3
> opt_sales3 <- arima(transData_ts, order=c(0,1,1),seasonal = list(order = c(0,0,0), period=12))
> opt_sales3$aic
[1] 85.55954
> tsdisplay(residuals(opt_sales3), lag.max=11,main ='Trained Sales Model')
> #predicted values for arima model 3
> opt_sales_forecast3<- forecast(opt_sales3, h=12)
> opt_sales_forecast3
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Feb 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Mar 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Apr 2011       17.65154 9.000941 26.30213 4.421592 30.88148
May 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Jun 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Jul 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Aug 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Sep 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Oct 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Nov 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Dec 2011       17.65154 9.000941 26.30213 4.421592 30.88148
Jan 2012       17.65154 9.000941 26.30213 4.421592 30.88148
> #plotting predicted values
> plot.ts(transData_ts)
> plot(opt_sales_forecast3)
```





```
> #ARIMA model 4
> opt_sales4 <- arima(transData_ts, order=c(1,1,1))
> opt_sales4$aic
[1] 86.79442
> tsdisplay(residuals(opt_sales4), lag.max=11,main ='Trained Sales Model')
> #predicted values for arima model 4
> opt_sales_forecast4<- forecast(opt_sales4, h=12)
> opt_sales_forecast4
         Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
Feb 2011       18.45276 10.229771 26.67575 5.876782 31.02874
Mar 2011       17.55364  9.226129 25.88115 4.817811 30.28947
Apr 2011       17.77803  9.406979 26.14908 4.975612 30.58045
May 2011       17.72203  9.357509 26.08655 4.929599 30.51446
Jun 2011       17.73601  9.369586 26.10243 4.940670 30.53134
Jul 2011       17.73252  9.366555 26.09848 4.937882 30.52715
Aug 2011       17.73339  9.367310 26.09947 4.938576 30.52820
Sep 2011       17.73317  9.367122 26.09922 4.938403 30.52794
Oct 2011       17.73323  9.367169 26.09928 4.938446 30.52800
Nov 2011       17.73321  9.367157 26.09927 4.938435 30.52799
Dec 2011       17.73322  9.367160 26.09927 4.938438 30.52799
Jan 2012       17.73321  9.367159 26.09927 4.938437 30.52799
> #plotting predicted values
> plot.ts(transData_ts)
> plot(opt_sales_forecast4)
```





Task 3: Compare how algorithms perform. Which one is better, why?

Out of the four Autoregressive Integrated Moving Average (ARIMA) models that I created the second ARIMA model performed the best. The second ARIMA model had its p parameter equal to 1 (specifying the number of lags used in the model), d parameter equal to 1 (representing the degree of differencing in the integrated component), and q parameter equal to 0 (representing the error of the model as a combination of previous error terms). The second model also has an order of seasonal differencing argument.

In the ACF plot of the second ARIMA model all of the model's spikes were not out of the 95% significance boundaries. As well as in the PACF plot, all of the time series spikes were not out of the 95% significance boundaries, too. However out of the four ARIMA models the third model also had the lowest AIC score of 85.56. Which generally the model that hast the lowest AIC score is the better model. However, the third model fails to incorporate a seasonal differencing term, which the original model residuals suggests. So, I still would suggest ARIMA model two over model three.

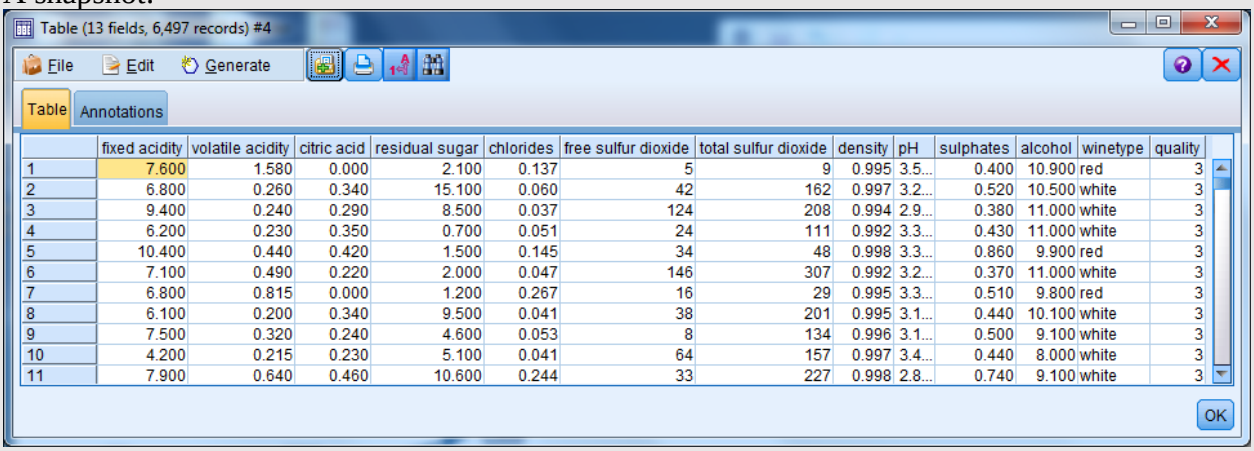
Problem 2 (hope you still remember it!) - Wine quality classification problem

The second problem in this final exam uses data stored in the following file *Q2_winequality.csv*. The following table provides details about the dataset.

Input variables (based on physicochemical tests)	
Fixed acidity	Numeric
Volatile acidity	Numeric
Citric acid	Numeric

Residual sugar	Numeric
Chlorides	Numeric
Free sulfur dioxide	Numeric
Total sulfur dioxide	Numeric
Density	Numeric
pH	Numeric
Sulphates	Numeric
Alcohol	Numeric (%)
Wine Type	red or white
Output variable (based on sensory data)	
Quality	Score between 0 and 10 in ordinal

A snapshot:



	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	winetype	quality
1	7.600	1.580	0.000	2.100	0.137	5	9	0.995	3.5...	0.400	10.900	red	3
2	6.800	0.260	0.340	15.100	0.060	42	162	0.997	3.2...	0.520	10.500	white	3
3	9.400	0.240	0.290	8.500	0.037	124	208	0.994	2.9...	0.380	11.000	white	3
4	6.200	0.230	0.350	0.700	0.051	24	111	0.992	3.3...	0.430	11.000	white	3
5	10.400	0.440	0.420	1.500	0.145	34	48	0.998	3.3...	0.860	9.900	red	3
6	7.100	0.490	0.220	2.000	0.047	146	307	0.992	3.2...	0.370	11.000	white	3
7	6.800	0.815	0.000	1.200	0.267	16	29	0.995	3.3...	0.510	9.800	red	3
8	6.100	0.200	0.340	9.500	0.041	38	201	0.995	3.1...	0.440	10.100	white	3
9	7.500	0.320	0.240	4.600	0.053	8	134	0.996	3.1...	0.500	9.100	white	3
10	4.200	0.215	0.230	5.100	0.041	64	157	0.997	3.4...	0.440	8.000	white	3
11	7.900	0.640	0.460	10.600	0.244	33	227	0.998	2.8...	0.740	9.100	white	3

```
> #Question 2.)
> #Dataset 2 - importing data for this question
> wineData<-read.csv("C:/Users/esbro/Desktop/IE 575/Final_Exam/Q2_winequality(1).csv")
> summary(wineData) #statistical summary of data from excel file
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides       free.sulfur.dioxide
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   :  1.00
 1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800   1st Qu.:0.03800   1st Qu.: 17.00
 Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000   Median :0.04700   Median : 29.00
 Mean   : 7.215   Mean   :0.3397   Mean   :0.3186   Mean   : 5.443   Mean   :0.05603   Mean   : 30.52
 3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100   3rd Qu.:0.06500   3rd Qu.: 41.00
 Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800   Max.   :0.61100   Max.   :289.00
 total.sulfur.dioxide    density             pH           sulphates         alcohol          winetype            quality
 Min.   :  6.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Length:6497        Min.   :3.000
 1st Qu.: 77.0        1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   Class :character   1st Qu.:5.000
 Median :118.0        Median :0.9949   Median :3.210   Median :0.5100   Median :10.30   Mode  :character   Median :6.000
 Mean   :115.7        Mean   :0.9947   Mean   :3.219   Mean   :0.5313   Mean   :10.49                      Mean   :5.818
 3rd Qu.:156.0        3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30                      3rd Qu.:6.000
 Max.   :440.0        Max.   :1.0390   Max.   :4.010   Max.   :2.0000   Max.   :14.90                      Max.   :9.000
> colSums(is.na(wineData)) #how many columns/variables have missing values
       fixed.acidity     volatile.acidity          citric.acid       residual.sugar            chlorides
                   0                    0                    0                    0                    0
 free.sulfur.dioxide total.sulfur.dioxide              density                   pH            sulphates
                   0                    0                    0                    0                    0
             alcohol             winetype              quality
                   0                    0                    0
```

(Note that you can simply copy or revise your midterm solution to Logistic Regression and CART models.)

Task 1: Build classification models including Logistic Regression, CART, C5.0, Neural Networks, and Bayesian Networks, and find an optimal classification model to predict wine quality for the problem on hand. Please use the split ratio, 0.7/0.3 (training/test), to train and test the chosen models.

```
> wineData$winetype<-as.factor(wineData$winetype)
> wineData$winetype<-as.numeric(wineData$winetype)
> wineData$quality<-factor(wineData$quality)
> #getting partition of training and test dataset correct - ratio 70:30 for Bayesian model
> library(caret)
Loading required package: ggplot2
Loading required package: lattice
Warning message:
package 'caret' was built under R version 4.1.3
> nb_sampling_vector <- createDataPartition(wineData$quality, p = 0.70, list = FALSE)
> training_Data <- wineData[nb_sampling_vector,]
```

```
> #Neural network model
> library("scales") #have to scale values between 0 and 1 for this model to work properly
> wineDataScaled<-wineData
> wineDataScaled$quality<-as.numeric(wineDataScaled$quality)
> wineDataScaled<-as.data.frame(sapply(wineDataScaled,function(x)rescale(x)))
> library(neuralnet)
Warning message:
package 'neuralnet' was built under R version 4.1.3
> set.seed(244)
> nb_sampling_vector2 <- createDataPartition(wineDataScaled$quality, p = 0.70, list = FALSE)
> training_Data2 <- wineDataScaled[nb_sampling_vector2,]
```

```
> test_Data2 <- wineDataScaled[-nb_sampling_vector2,]
> nn <- neuralnet(f, data = training_Data2,hidden = 6,linear.output=FALSE, learningrate = 0.05, threshold = 0.1)
> #prediction values and results from the neural network model on the test dataset
> predicted.nn.values <- compute(nn,test_Data2)
> print(head(predicted.nn.values$net.result))
```

```
       [,1]
3  0.4101681
5  0.3448232
6  0.3586171
9  0.3906854
17 0.4388972
21 0.4133102
> predicted.nn.values$net.result <- sapply(predicted.nn.values$net.result,round,digits=0)
> #accuracy=(true positives + true Negatives)/(true Positives+true Negatives+false Positives+false Negatives)
> cm<-table(test_Data2$quality,predicted.nn.values$net.result)
> cm

                   0   1
  0                11   2
  0.166666666666667  63   8
  0.333333333333333 573  58
  0.5               526 324
  0.666666666666667  82 245
  0.833333333333333   5  49
  1                   0   2
> accuracyResult<-sum(diag(cm))/sum(cm)
> accuracyResult
[1] 0.009753593
> #Building the CART classification model
> #to build a decision tree based on CART, we can use rpart function
> set.seed(2448)
> library(rpart)
Warning message:
package 'rpart' was built under R version 4.1.3
> cart_qualMod<- rpart(f, method="class", data=training_Data, cp = 0.001)
```

```
> test_Data <- wineData[-nb_sampling_vector,]
> #formulated model to use for each of classification models used in this problem
> feats <- names(wineData[1:12])
> # Concatenate strings
> f <- paste(feats,collapse=' + ')
> f <- paste('quality ~',f)
> # Convert to formula
> f <- as.formula(f)
> f
quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol + winetype
> #Bayesian model
> set.seed(2448)
> library("e1071")
Warning message:
package 'e1071' was built under R version 4.1.3
> training_nb_model<-naiveBayes(f,data=training_Data)
> nb_test_predictions<- predict(training_nb_model,test_Data)
> #getting the native Bayes model results
> mean(nb_test_predictions==test_Data$quality) #accuracy results from the model
[1] 0.4107969
> table(actual=test_Data$quality,predictions=nb_test_predictions)
      predictions
actual   3   4   5   6   7   8   9
     3   2   1   1   2   3   0   0
     4   1   7  25  22   8   1   0
     5  11  22 294 255  56   0   3
     6  11  14 214 350 225   1  35
     7   5   4  28  99 143   7  37
     8   0   1   3  13  32   3   5
     9   0   0   0   1   0   0   0
> #use printcp() function to show the table of complexity parameter, including the cross-validated error
> printcp(cart_qualMod)

Classification tree:
rpart(formula = f, data = training_Data, method = "class", cp = 0.001)

Variables actually used in tree construction:
 [1] alcohol             chlorides           citric.acid         density             fixed.acidity
 [6] free.sulfur.dioxide pH                  residual.sugar      sulphates           total.sulfur.dioxide
[11] volatile.acidity

Root node error: 2566/4552 = 0.56371

n= 4552

          CP nsplit rel error  xerror     xstd
1  0.1071707      0   1.00000 1.00000 0.013039
2  0.0596259      1   0.89283 0.89205 0.013146
3  0.0075994      2   0.83320 0.84100 0.013129
4  0.0049363      4   0.81800 0.83087 0.013120
5  0.0035074      7   0.80320 0.83398 0.013123
6  0.0033125      8   0.79969 0.81567 0.013104
7  0.0032476     10   0.79306 0.81567 0.013104
8  0.0031177     17   0.76345 0.81723 0.013106
9  0.0027280     18   0.76033 0.80592 0.013092
10 0.0023383     20   0.75487 0.80359 0.013088
11 0.0021434     32   0.72642 0.80553 0.013091
12 0.0020460     34   0.72214 0.80865 0.013095
13 0.0019486     39   0.71161 0.81021 0.013097
14 0.0018024     43   0.70382 0.81956 0.013109
15 0.0017147     53   0.68394 0.82190 0.013111
16 0.0016368     59   0.67303 0.82424 0.013114
17 0.0015588     66   0.66134 0.82424 0.013114
18 0.0014030     88   0.62120 0.82151 0.013111
19 0.0013640     94   0.61263 0.82424 0.013114
20 0.0012990    100   0.60444 0.82385 0.013113
21 0.0011691    107   0.59470 0.82385 0.013113
22 0.0010717    143   0.54754 0.81995 0.013109
23 0.0010392    149   0.54053 0.82307 0.013112
24 0.0010000    153   0.53624 0.82385 0.013113
```

```
> #Calculate the accuracy of the predicted tree based on CART model
> q2_predictedCart<-predict(cart_qualMod,newdata=test_Data,type="class")
> mean(test_Data$quality==q2_predictedCart)
[1] 0.5511568
> #building ordinal logistic regression model
> library(MASS)
> set.seed(16)
> q2Data.glm<- polr(f,
+                   data = training_Data, Hess=TRUE)
> #informative details of ordinal logistic regression model
> summary(q2Data.glm)
Call:
polr(formula = f, data = training_Data, Hess = TRUE)

Coefficients:
                        Value Std. Error  t value
fixed.acidity         2.410e-01   0.032958    7.312
volatile.acidity     -4.365e+00   0.273444  -15.964
citric.acid          -4.890e-01   0.254462   -1.922
residual.sugar        1.679e-01   0.007714   21.762
chlorides            -1.929e+00   1.080312   -1.786
free.sulfur.dioxide   1.697e-02   0.002474    6.858
total.sulfur.dioxide -3.909e-03   0.001006   -3.888
density              -2.943e+02   0.539506 -545.498
pH                    1.399e+00   0.228307    6.130
sulphates             2.305e+00   0.235544    9.786
alcohol               5.910e-01   0.032108   18.406
winetype             -1.004e+00   0.153173   -6.555

Intercepts:
    Value     Std. Error t value
3|4 -287.7922    0.5467  -526.4294
4|5 -285.5668    0.5443  -524.6220
5|6 -282.3454    0.5472  -515.9794
6|7 -279.7303    0.5557  -503.3943
7|8 -277.3953    0.5653  -490.6937
8|9 -273.7474    0.7509  -364.5349

Residual Deviance: 9871.728
AIC: 9907.728
```

```
> #computing the model's probability predictions for the observations in the wine dataset
> prediction_LogOrd_Wine<- predict(q2Data.glm, newdata = test_Data, type = "class")
> head(prediction_LogOrd_Wine)
[1] 5 5 5 6 5 6
Levels: 3 4 5 6 7 8 9
> #now getting the prediction performances of the logistic regression model on wine dataset
> #making confusion matrix
> cm1<-as.matrix(table(Actual=test_Data$quality,Predicted=prediction_LogOrd_Wine))
> cm1
      Predicted
Actual   3   4   5   6   7   8   9
     3   0   0   4   4   1   0   0
     4   0   1  38  25   0   0   0
     5   0   1 370 268   2   0   0
     6   0   0 203 590  54   3   0
     7   0   0  16 227  80   0   0
     8   0   0   3  37  17   0   0
     9   0   0   0   1   0   0   0
> #accuracy=(true positives + true Negatives)/(true Positives+true Negatives+false Positives+false Negatives)
> accuracyResult<-sum(diag(cm1))/sum(cm1)
> accuracyResult
[1] 0.5352185
> #C5.0 classification decision tree model
> library(C50)
Warning message:
package 'C50' was built under R version 4.1.3
> set.seed(559)
```

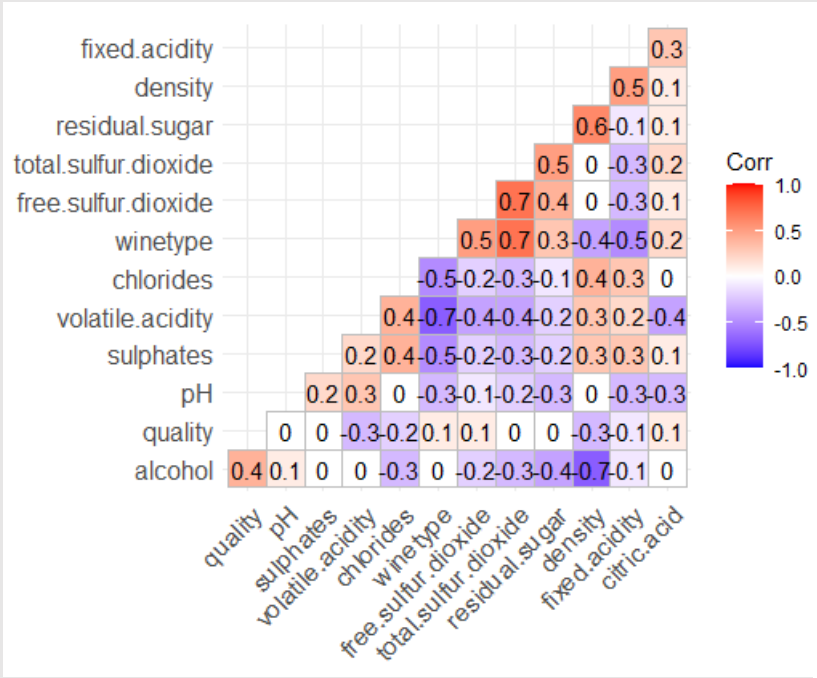
```
> wineQuality_C5<- C5.0(f, data=training_Data)
#algorithm computation results
summary(wineQuality_C5)

Evaluation on training data (4552 cases):

          Decision Tree
        ----------------
        Size      Errors

         668  540(11.9%)   <<


         (a)   (b)   (c)   (d)   (e)   (f)   (g)    <-classified as
        ----  ----  ----  ----  ----  ----  ----
          13     1     3     2     1     1          (a): class 3
           2   111    13    25     1                (b): class 4
           2    14  1311   163     3     4          (c): class 5
                 4   110  1828    39     5          (d): class 6
           1     5    23    84   639     4          (e): class 7
                      8     7    13   108          (f): class 8
                            1           1     2    (g): class 9


        Attribute usage:

        100.00% volatile.acidity
        100.00% alcohol
         82.43% residual.sugar
         79.48% free.sulfur.dioxide
         74.76% pH
         71.13% sulphates
         66.32% fixed.acidity
         52.42% chlorides
         45.58% total.sulfur.dioxide
         44.77% citric.acid
         24.63% winetype
         19.97% density


Time: 0.1 secs
```

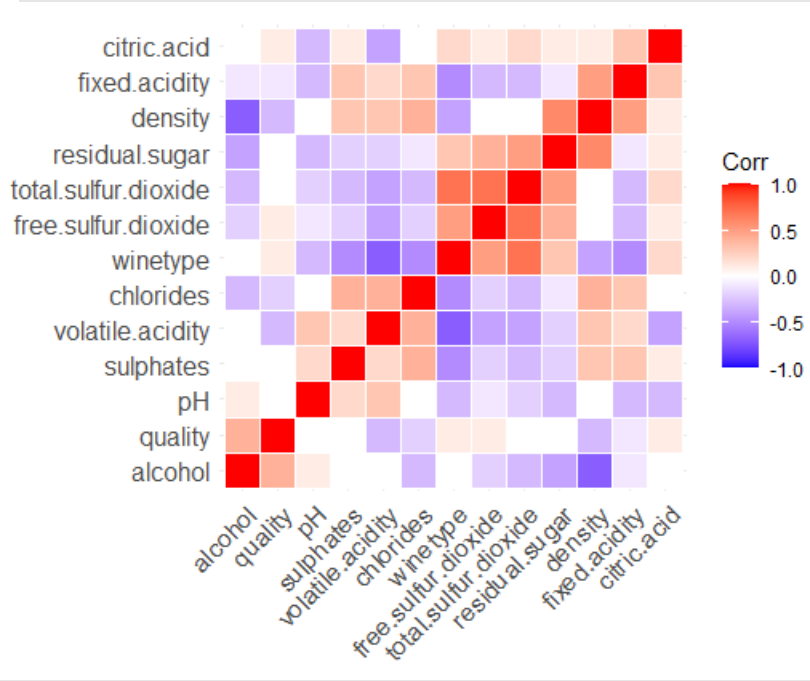
```
> wineQuality_C5<- C5.0(f, data=training_Data)
> #accuracy results from C5.0 model
> test_predictions<- predict(wineQuality_C5, test_Data)
> table(predicted = test_predictions, actual = test_Data$quality)
         actual
predicted   3   4   5   6   7   8   9
        3   1   0   1   3   0   0   0
        4   2   6  27  10   2   0   0
        5   1  29 386 172  26   0   0
        6   4  26 201 545 131  18   1
        7   1   1  23 105 154  19   0
        8   0   2   3  15  10  20   0
        9   0   0   0   0   0   0   0
> mean(test_predictions==test_Data$quality)
[1] 0.5717224
>
```

Task 2: By exploring the dataset, please identify if there is a simple and practical method to improve the accuracy of the wine quality prediction.

```
> #Question 2 - part 2.)
> #getting correlation results of each variable in the original wine dataset
> #install.packages("ggcorrplot")
> library(ggcorrplot)
Warning message:
package 'ggcorrplot' was built under R version 4.1.3
> corr <- round(cor(wineDataScaled), 1)
> head(corr[, 1:6])
                      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides free.sulfur.dioxide
fixed.acidity                   1.0              0.2         0.3            -0.1       0.3                -0.3
volatile.acidity                0.2              1.0        -0.4            -0.2       0.4                -0.4
citric.acid                     0.3             -0.4         1.0             0.1       0.0                 0.1
residual.sugar                 -0.1             -0.2         0.1             1.0      -0.1                 0.4
chlorides                       0.3              0.4         0.0            -0.1       1.0                -0.2
free.sulfur.dioxide            -0.3             -0.4         0.1             0.4      -0.2                 1.0
> #outputting these correlation results in a matrix
> ggcorrplot(corr,
+            hc.order = TRUE,
+            type = "lower",
+            lab = TRUE)
> ggcorrplot(corr, hc.order = TRUE, outline.color = "white")
```

Correlation Matrix of Wine Data – Variable Values Scaled
(between 0 & 1)





From the correlation matrix of the wine data variables, we can see that there are four variables that are somewhat correlated to quality. Those variables are alcohol, volatile.acidity, chlorides, and density. The other variables are less than 0.2 percent correlated to a wine's quality. So, to help improve the accuracy of the wine quality prediction, we should get rid of those variables

that are less than 0.2 percent correlated to wine's quality. And keep only the four variables that are greater or equal to 0.2 percent correlated to a wine's quality score.

```
> #Bayesian model
> set.seed(2448)
> training_nb_model_Pt2<-naiveBayes(f_Pt2,data=training_Data_Pt2)
> nb_test_predictions_Pt2<- predict(training_nb_model_Pt2,test_Data_Pt2)
> #getting the native Bayes model accuracy results
> mean(nb_test_predictions_Pt2==test_Data_Pt2$quality) #accuracy results from the model
[1] 0.474036
> #neural network model
> wineDataScaled_Pt2<-wineData2
> wineDataScaled_Pt2$quality<-as.numeric(wineDataScaled_Pt2$quality)
> wineDataScaled_Pt2<-as.data.frame(sapply(wineDataScaled_Pt2,function(x)rescale(x))) #scaling variable values between 0 & 1
> set.seed(244)
> nb_sampling_vector_scaled_Pt2 <- createDataPartition(wineDataScaled_Pt2$quality, p = 0.70, list = FALSE)
> training_Data_scaled <- wineDataScaled_Pt2[nb_sampling_vector_scaled_Pt2,]
> test_Data_scaled <- wineDataScaled_Pt2[-nb_sampling_vector_scaled_Pt2,]
> nn_Pt2 <- neuralnet(f_Pt2, data = training_Data_scaled,hidden = 2,linear.output=FALSE, learningrate = 0.05, threshold = 0.1)
> predicted.nn.values_Pt2 <- compute(nn_Pt2,test_Data_scaled)
> print(head(predicted.nn.values_Pt2$net.result))
       [,1]
3  0.3397024
5  0.3367054
6  0.3453055
9  0.3611839
17 0.4898883
21 0.4392049
> predicted.nn.values_Pt2$net.result <- sapply(predicted.nn.values_Pt2$net.result,round,digits=0)
> #accuracy=(true positives + true Negatives)/(true Positives+true Negatives+false Positives+false Negatives)
> cm_Pt2<-table(test_Data_scaled$quality,predicted.nn.values_Pt2$net.result)
> cm_Pt2

                    0   1
  0                10   3
  0.166666666666667  57  14
  0.333333333333333 556  75
  0.5               517 333
  0.666666666666667  92 235
  0.833333333333333   5  49
  1                   1   1
```

```
> accuracyResult_Pt2<-sum(diag(cm_Pt2))/sum(cm_Pt2)
> accuracyResult_Pt2
[1] 0.01232033
> #Building the CART classification model
> #to build a decision tree based on CART, we can use rpart function
> set.seed(2448)
> library(rpart)
> cart_qualMod_Pt2<- rpart(f_Pt2, method="class", data=training_Data_Pt2, cp = 0.001)
> #use printcp() function to show the table of complexity parameter, including the cross-validated error
> printcp(cart_qualMod_Pt2)

Classification tree:
rpart(formula = f_Pt2, data = training_Data_Pt2, method = "class",
    cp = 0.001)

Variables actually used in tree construction:
[1] alcohol          chlorides        density          volatile.acidity

Root node error: 2566/4552 = 0.56371

n= 4552
```

```
> #only including variables that are the most correlated to the outcome variable quality
> #reordering the dataset to create new formula to model based on top variables correlated to wine's quality
> wineData2 <- wineData[, c(11,2,5,8,13)]
> feats2 <- names(wineData2[1:4])
> # Concatenate strings
> f_Pt2 <- paste(feats2,collapse=' + ')
> f_Pt2 <- paste('quality ~',f_Pt2)
> # Convert to formula
> f_Pt2 <- as.formula(f_Pt2)
> f_Pt2
quality ~ alcohol + volatile.acidity + chlorides + density
> #splitting revised dataset with the most highly correlated variables into training and test datasets ratio 70:30
> nb_sampling_vector_Pt2 <- createDataPartition(wineData2$quality, p = 0.70, list = FALSE)
> training_Data_Pt2 <- wineData2[nb_sampling_vector_Pt2,]
> test_Data_Pt2 <- wineData2[-nb_sampling_vector_Pt2,]
>
```

```
          CP nsplit rel error  xerror     xstd
1  0.1060016      0   1.00000 1.00000 0.013039
2  0.0678098      1   0.89400 0.91660 0.013139
3  0.0101325      2   0.82619 0.84528 0.013132
4  0.0064302      4   0.80592 0.82931 0.013119
5  0.0062354      6   0.79306 0.82073 0.013110
6  0.0050663      7   0.78683 0.81333 0.013101
7  0.0027280      8   0.78176 0.80086 0.013085
8  0.0023383      9   0.77903 0.80047 0.013084
9  0.0019486     10   0.77670 0.79657 0.013078
10 0.0017537     14   0.76773 0.79462 0.013075
11 0.0015588     20   0.75604 0.79696 0.013079
12 0.0013640     31   0.73772 0.80047 0.013084
13 0.0013362     33   0.73500 0.80047 0.013084
14 0.0011691     49   0.71005 0.80164 0.013086
15 0.0010825     67   0.68628 0.80904 0.013096
16 0.0010000     77   0.67537 0.81255 0.013100
> #Calculate the accuracy of the predicted tree based on CART model
> q2_predictedCart_Pt2<-predict(cart_qualMod_Pt2,newdata=test_Data_Pt2,type="class")
> mean(test_Data_Pt2$quality==q2_predictedCart_Pt2)
[1] 0.5280206
> #building ordinal logistic regression model
> set.seed(16)
> library(MASS)
> q2Data.glm_Pt2<- polr(f_Pt2,
+                  data = training_Data_Pt2, Hess=TRUE)
> #informative details of ordinal logistic regression model
> summary(q2Data.glm_Pt2)
Call:
polr(formula = f_Pt2, data = training_Data_Pt2, Hess = TRUE)

Coefficients:
                  Value Std. Error  t value
alcohol           1.019    0.03716  27.4189
volatile.acidity -3.667    0.20789 -17.6370
chlorides        -0.467    0.95481  -0.4891
density          94.474   14.25863   6.6258
```

```
wineQuality_C5_Pt2<- C5.0(f_Pt2, data=training_Data_Pt2)
#algorithm computation results
summary(wineQuality_C5_Pt2)
```

```
Evaluation on training data (4552 cases):

           Decision Tree
         ----------------
         Size      Errors

          578  984(21.6%)   <<


          (a)   (b)   (c)   (d)   (e)   (f)   (g)    <-classified as
         ----  ----  ----  ----  ----  ----  ----
            9           4     5     2     1          (a): class 3
            1    65    49    34     2     1          (b): class 4
            2    11  1230   232    20     2          (c): class 5
            1     8   235  1654    81     7          (d): class 6
            1     3    26   186   533     7          (e): class 7
                  4     5    31    19    77          (f): class 8
                              2     2                (g): class 9


        Attribute usage:

        100.00% alcohol
        100.00% volatile.acidity
         72.67% chlorides
         72.06% density


Time: 0.0 secs
```

```
Intercepts:
    Value    Std. Error t value
3|4  97.3790  14.4082     6.7586
4|5  99.5542  14.4079     6.9097
5|6 102.6567  14.4144     7.1218
6|7 105.1912  14.4177     7.2960
7|8 107.4771  14.4190     7.4538
8|9 111.1058  14.4277     7.7009

Residual Deviance: 10112.45
AIC: 10132.45
> #computing the model's probability predictions for the observations in the wine dataset
> prediction_LogOrd_Wine_Pt2<- predict(q2Data.glm_Pt2, newdata = test_Data_Pt2, type = "class")
> head(prediction_LogOrd_Wine_Pt2)
[1] 6 5 6 5 5 5
Levels: 3 4 5 6 7 8 9
> #now getting the prediction performances of the logistic regression model on wine dataset
> #making confusion matrix
> cm1_Pt2<-as.matrix(table(Actual=test_Data_Pt2$quality,Predicted=prediction_LogOrd_Wine_Pt2))
> cm1_Pt2
      Predicted
Actual   3   4   5   6   7   8   9
     3   0   1   6   2   0   0   0
     4   0   0  40  22   2   0   0
     5   0   0 374 266   0   1   0
     6   0   0 204 596  50   0   0
     7   0   0  22 247  54   0   0
     8   0   0   3  37  17   0   0
     9   0   0   0   1   0   0   0
> #accuracy=(true positives + true Negatives)/(true Positives+true Negatives+false Positives+false Negatives)
> accuracyResult2_Pt2<-sum(diag(cm1_Pt2))/sum(cm1_Pt2)
> accuracyResult2_Pt2
[1] 0.5264781
> #C5.0 classification decision tree model
> library(C50)
> set.seed(559)
```

```
> #accuracy results from C5.0 model
> test_predictions_Pt2<- predict(wineQuality_C5_Pt2, test_Data_Pt2)
> table(predicted = test_predictions_Pt2, actual = test_Data_Pt2$quality)
         actual
predicted   3   4   5   6   7   8   9
        3   2   3   3   2   0   1   0
        4   0   6  10  10   3   0   0
        5   3  25 414 202  28   2   0
        6   4  29 184 532 155  24   1
        7   0   1  29  94 121  18   0
        8   0   0   1  10  16  12   0
        9   0   0   0   0   0   0   0
> mean(test_predictions_Pt2==test_Data_Pt2$quality)
[1] 0.5588689
```

Additional Response: So, as we can see most the classification models improved their accuracy scores of predicting a wine's quality grade based on the reduction of non-significant independent variables of original model. Since, we now had the model of quality ~ alcohol + volatile.acidity + chlorides + density. While this simple method of improving the classification model's prediction of a wine's quality was successful for two models (Neural Networks & Bayesian Networks), it was not perfect. Since the Logistic Regression, CART, and C5.0 model's accuracy score slightly reduced from the reduction of independent variables in the formulated model.