Project Report

|  |  |
| --- | --- |
| **Course Name (NICF)** | **NICF Diploma in Business Analytics** |
| Product Name (Marketing & Sales) | Professional Diploma in Data Science |
| **Module Name (NICF)** | **NICF Basic R Programming** |
| Product Name (Marketing & Sales) | Basic R Programming |

|  |  |  |  |
| --- | --- | --- | --- |
| Student name | | Assessor name | |
|  | |  | |
| Date issued | Completion date | | Submitted on |
|  |  | |  |
|  | |  | |
| Project title | Design and deploy Forecasting Model | | |

|  |
| --- |
| Learner declaration |
| I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.  Student signature: Date: |

Content

1. Project Overview 1
2. Project Technical Environment 2
3. Forecasting Model 3
4. Setting up the Forecasting Model 4
5. R Code for Data Filtering and Transformation 5
6. Transformed Data 6
7. Correlation Analysis Process and Output 7
8. Implementing Seasonal Forecasting Model 8
9. Refine Model 9
10. Forecast Output after Model Adjustment: 10
11. Project Overview: Describe the Project along with Project Outcomes (Explain the Project in your own words in 15 – 20 lines) **final outcome should be the outcome of script 3** columns

This project uses our knowledge of basic R Programming, including vectors, lists, factors and dataframes; and combines it with basic understanding of Azure Machine Learning to produce simple correlation analysis and 2 time series models which use the same datapoints but use a trend line or seasonal line to a clearer picture and from there analyse which forecasts are more accurate.

We have also learnt many new R functions along the way, which help in cleaning and transforming the data; and defining new functions that carry out statistical formulae.

Put together, we use 3 “Execute R Script”. 1 to cleanse and transform, another to perform correlation analysis and another to perform trend and forecasting analysis.

The 2nd script found there to be no significant correlations between the production of cheese, ice-cream, milk, and the price of milkfat(commodity).

The 3rd script gives us a trend model and a seasonal model, we have used data from 1995- 2012 to train our model and the data from the whole 12 months of 2013 for the forecast. We score the model based on the numbers shown below (final outcome of script 3).

From the result we can see that the forecast model and the training model are not too far apart in terms of RMS error and that the seasonal model is far more accurate than the trend model.

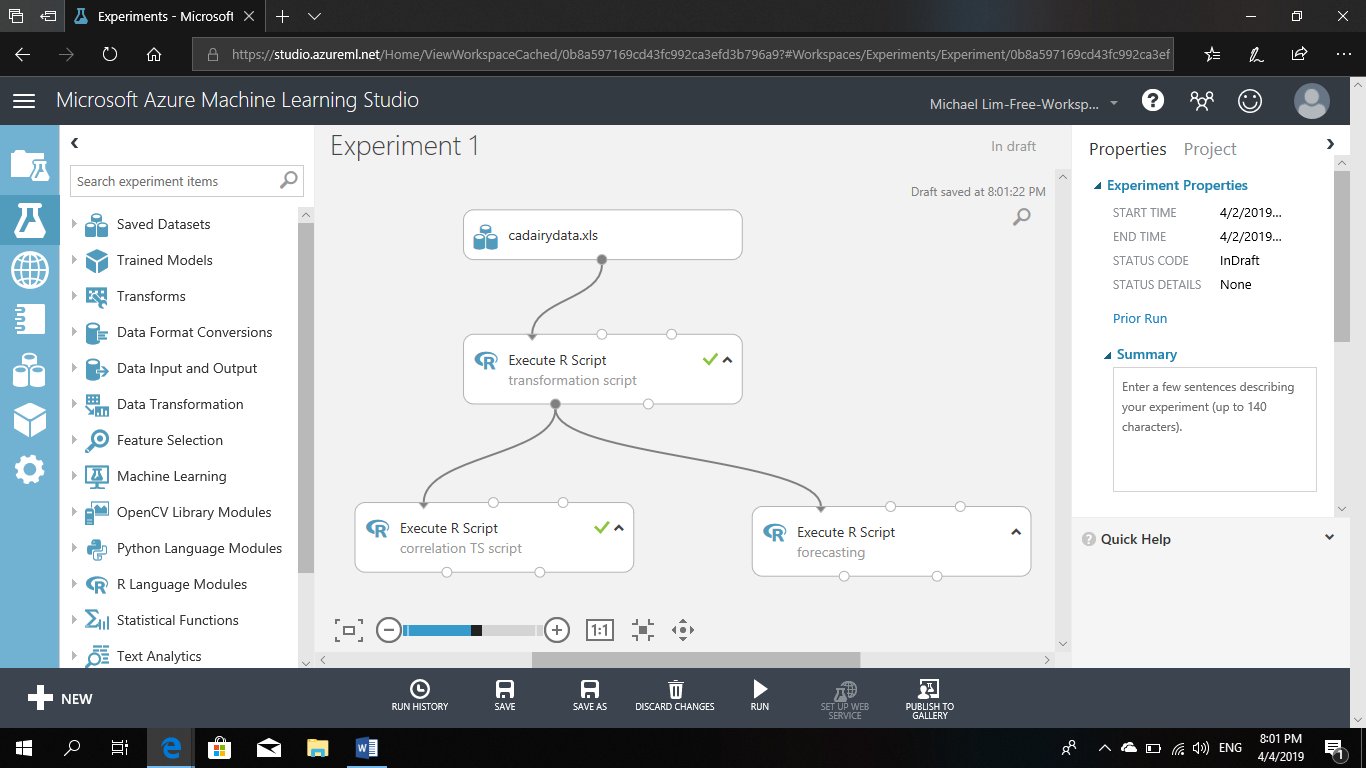
3

|  |  |  |  |
| --- | --- | --- | --- |
|  | rownames | Training | forecast |
| view as |  |  |  |
|  | Trend Model | 122.238008 | 174.088492 |
|  | Seasonal model | 75.115958 | 94.725325 |

1. Project Technical Environment: (Describe the MS AML Architecture with Tools used) import data, preprocessing, Split data, train model ,score model, algorithms , screen shot of the AML ( cadarirydata, script1,2,3)

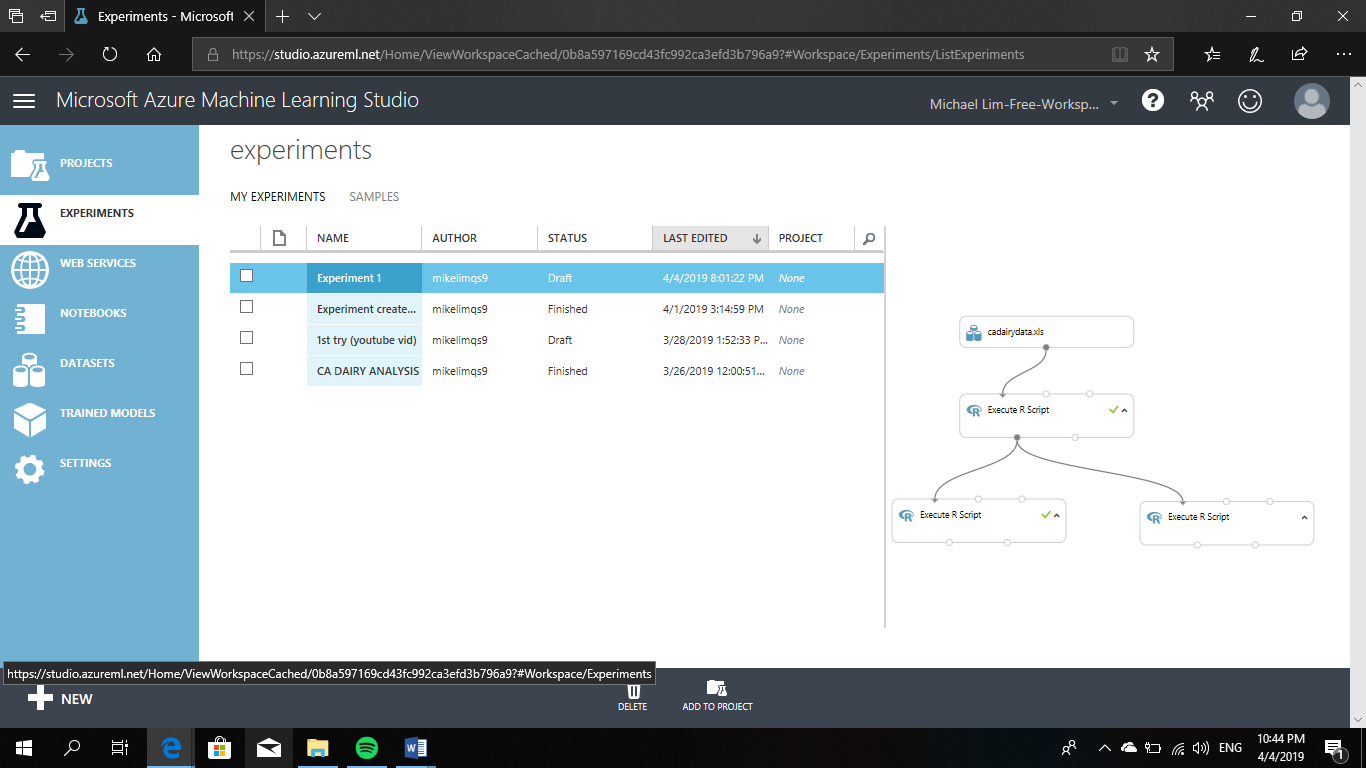
Microsoft Azure Machine Learning allows us to import data or use sample datasets, from which we can preform analysis using all sorts of algorithms like regression, classification, anomaly detection, text recognition and statistical functions. The usual process is to import data, clean and transform the data to prepare it for analysis, split the data (for training and scoring), and use built in algorithms to train the model then to score the model. From there we can click publish to gallery to make an API that clients or employers can use to view the analysis.

In our project, rather than using AML built in algorithms, we use Execute R Script modules to customize our own transformations and to perform correlation and forecasting.



1. Forecasting Model: (Explain the Forecasting model which you are designing and deploying on Azure AML)( steps of forecasting model)

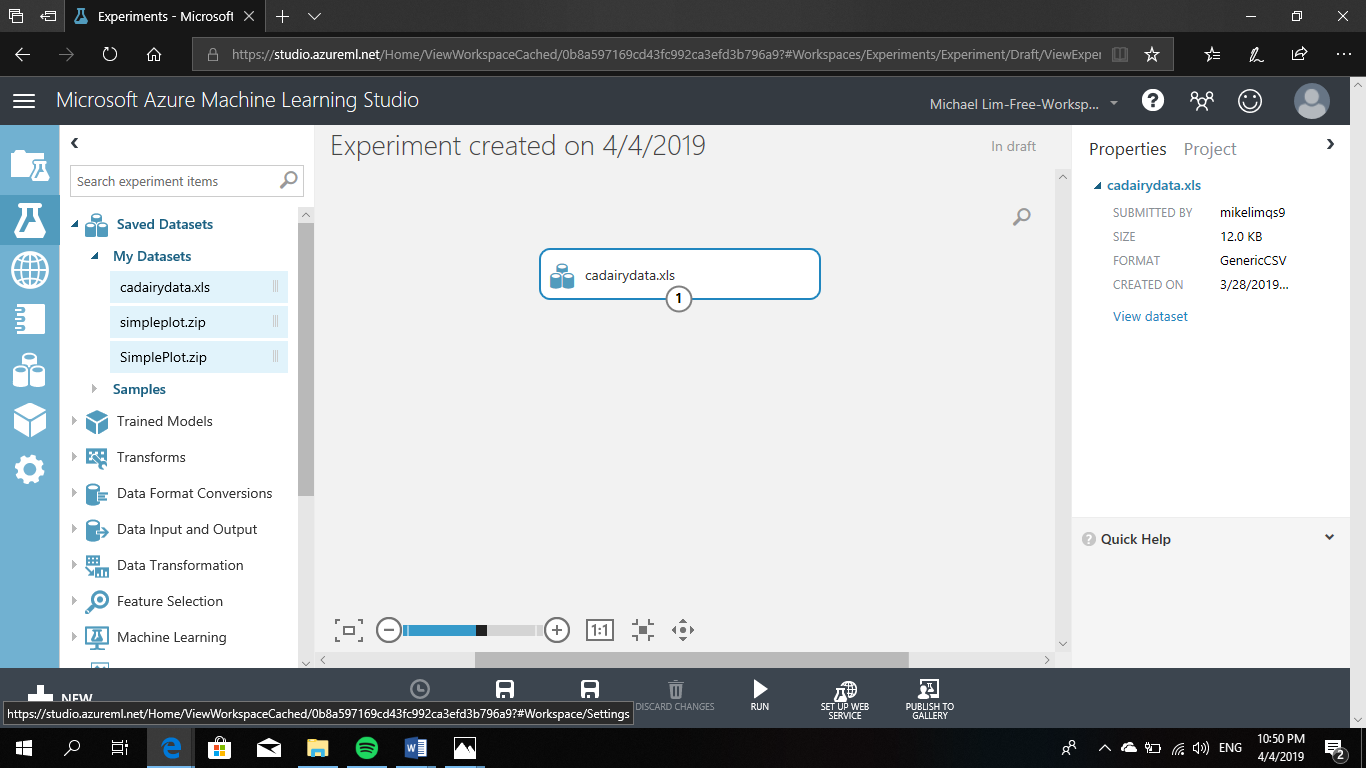
Step 1 : Load dataset



Click ‘New’ -> ‘Dataset’ -> ‘From Local File’

Upload cadairydata.xls.

Drag and drop into blank experiment.



After loading we can click the output port to visualize the data.

Step 2: Data preprocessing

Transformation:

* First, we remove 2 unneeded columns, 1 with the row numbers and another called $Year.Month which holds the same information found in $Year.
* Next, we categorize months as factors rather than characters.
* Then we create a new column that is going to be used for our timeseries models later on called month count. This counts months since the beginning of the timeseries.
* We then transform our different KPI to the same unit of measurement by using a list containing the conversion values.
* Lastly using the na.omit() function we omit any missing data.
* Data is now ready for correlation and trend analysis.

Step 3: Define time series , correlation analysis ( full script2)

* Create the time series using POSIXct and strptime
* Create scatterplots
* Detrend and standardize the variables
* Use the detrended variables to recreate scatterplots
* Use pairwise correlation analysis to check if there is any significant correlation. Lags are used to ensure that the variables are detrended.

Step 4: Train mode

A time series forecasting model will be created to forecast California milk production for the 12 months of 2013.

The forecasting model has the following characteristics:

* It has both the trend component and the seasonal component.

A training dataset is created with all of the observations except the last 12, of the year 2013, which is our test dataset, using this R expression.

cadairytrain <- cadairydata[1:216, ]

This training dataset is subsequently used for creating the trend and seasonal component for the time series forecasting model.

Step 5: Score model

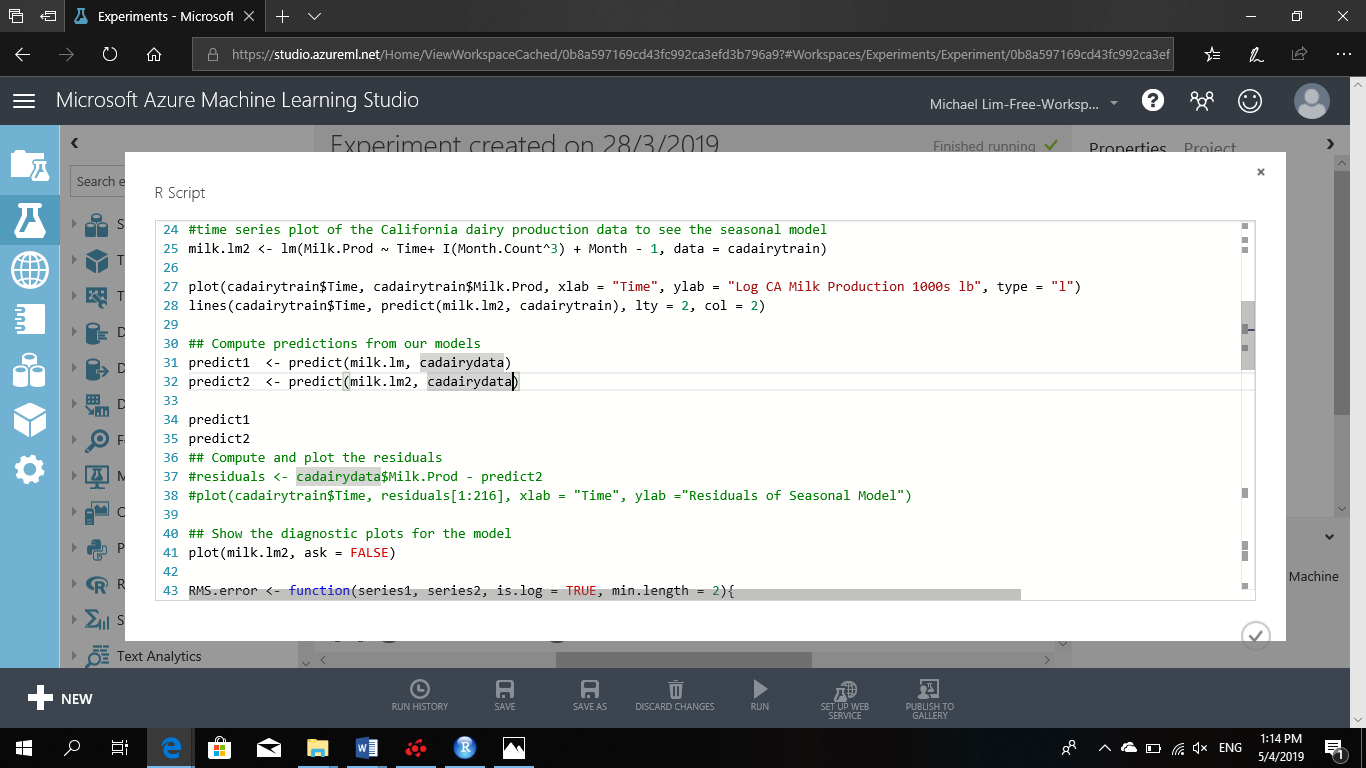
* In order to evaluate whether using a trend or seasonal component for the time series forecasting model is better, we use root mean square (RMS) error to measure the predicted dataset against the actual data that is not part of our training dataset. The smaller the RMS error, the better the forecast for the trend or seasonal forecasting model.

|  |  |  |
| --- | --- | --- |
| rownames | Training | forecast |
|  |  |  |
| Trend Model | 122.238008 | 174.088492 |
| Seasonal model | 75.115958 | 94.725325 |

*these are the figures for rms error.*

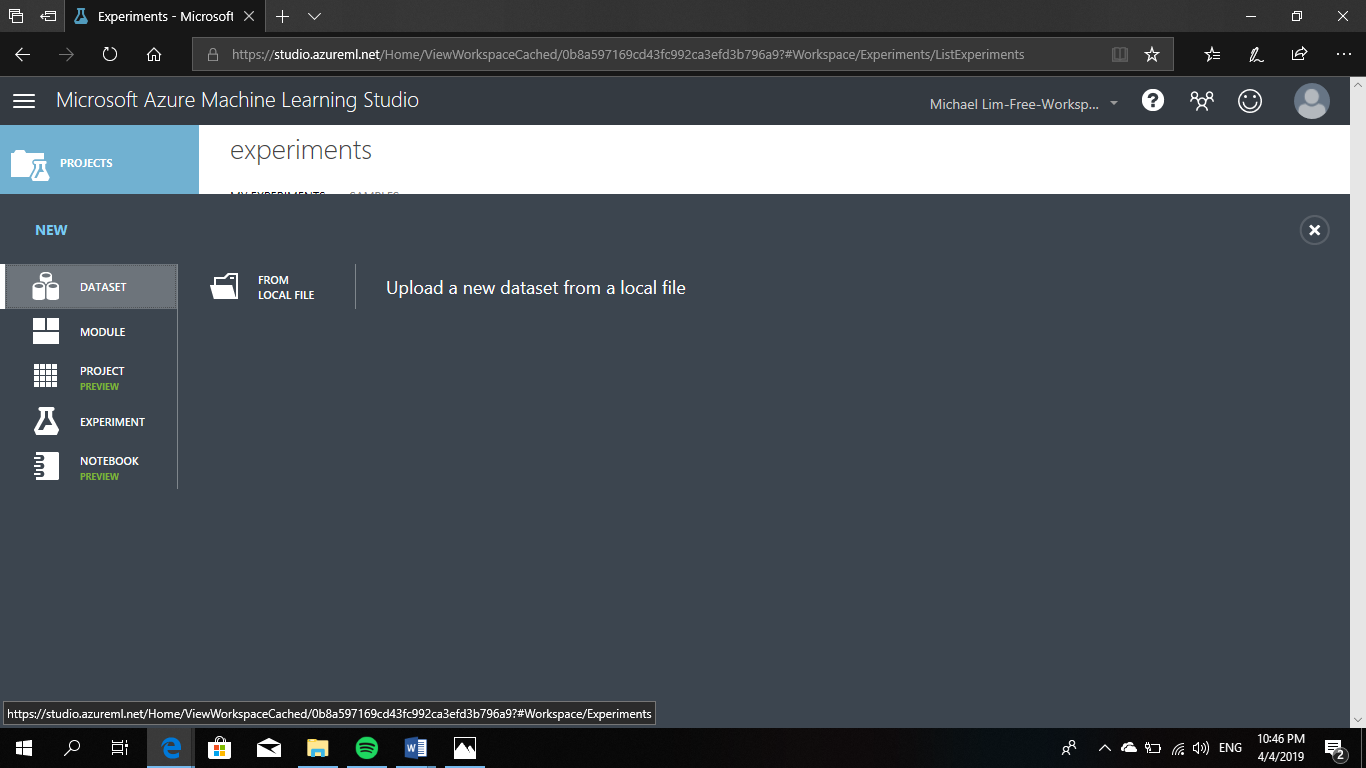
Step 6: Forecasting

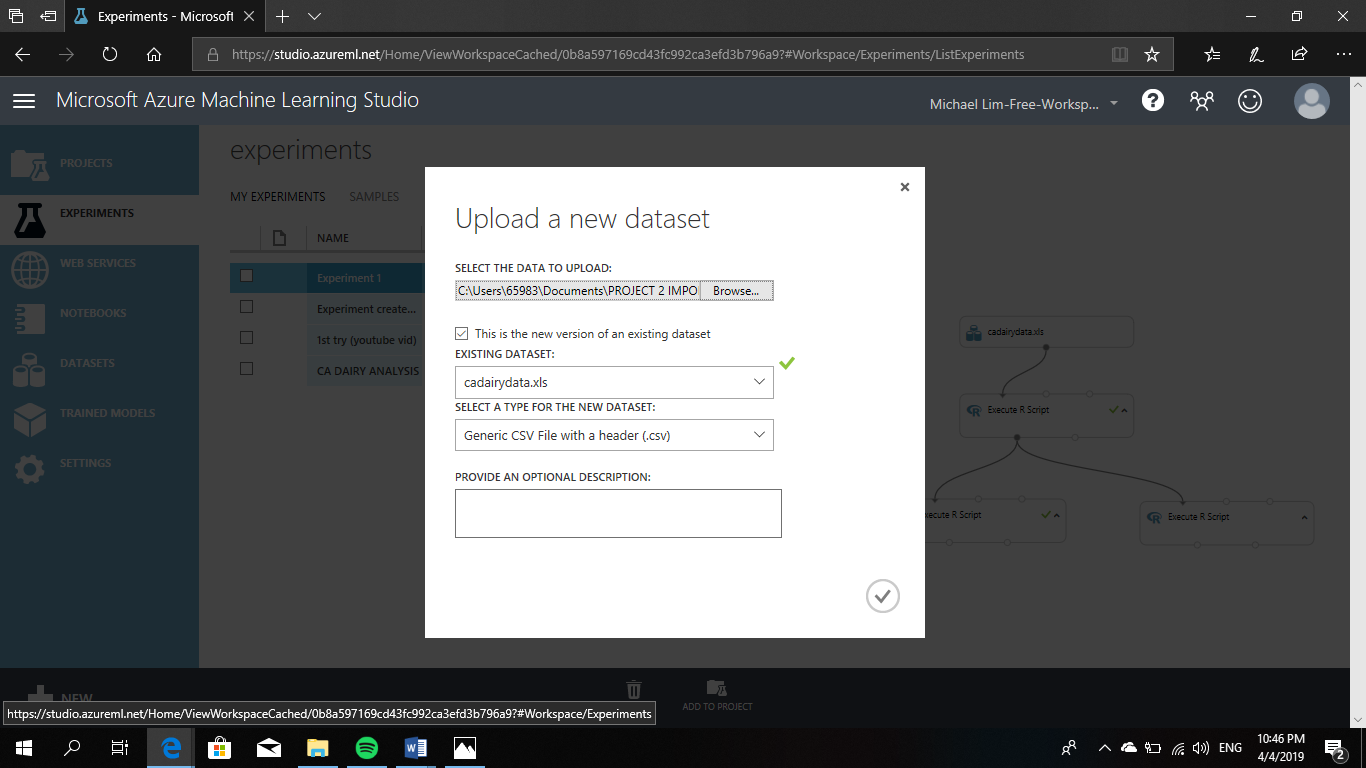
If we import a new dataframe with future values for $Time, and use it to change the portions highlighted in grey, it should be possible to come up with new forecasts for milk production. However this is not covered within the scope of this project.

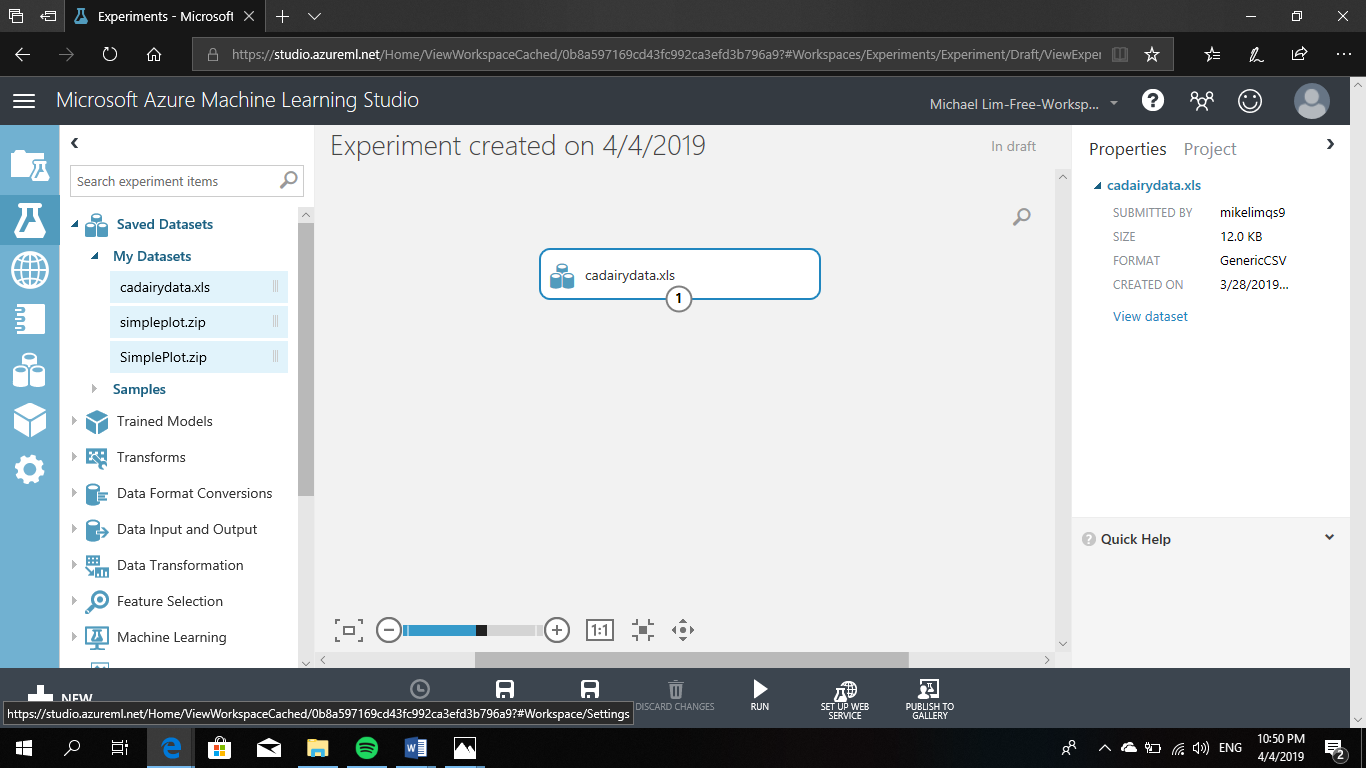


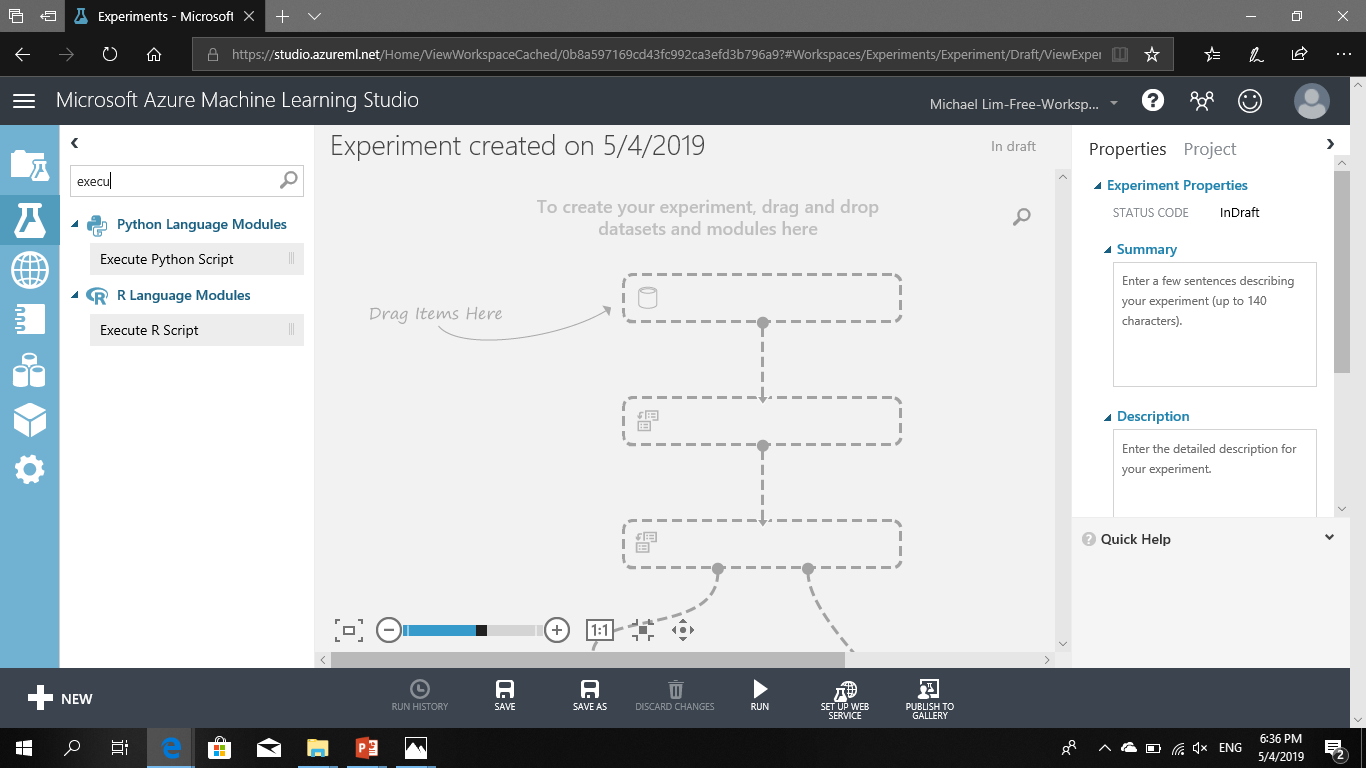
Instead we focus on training and scoring the model and reducing rms error to improve accuracy.

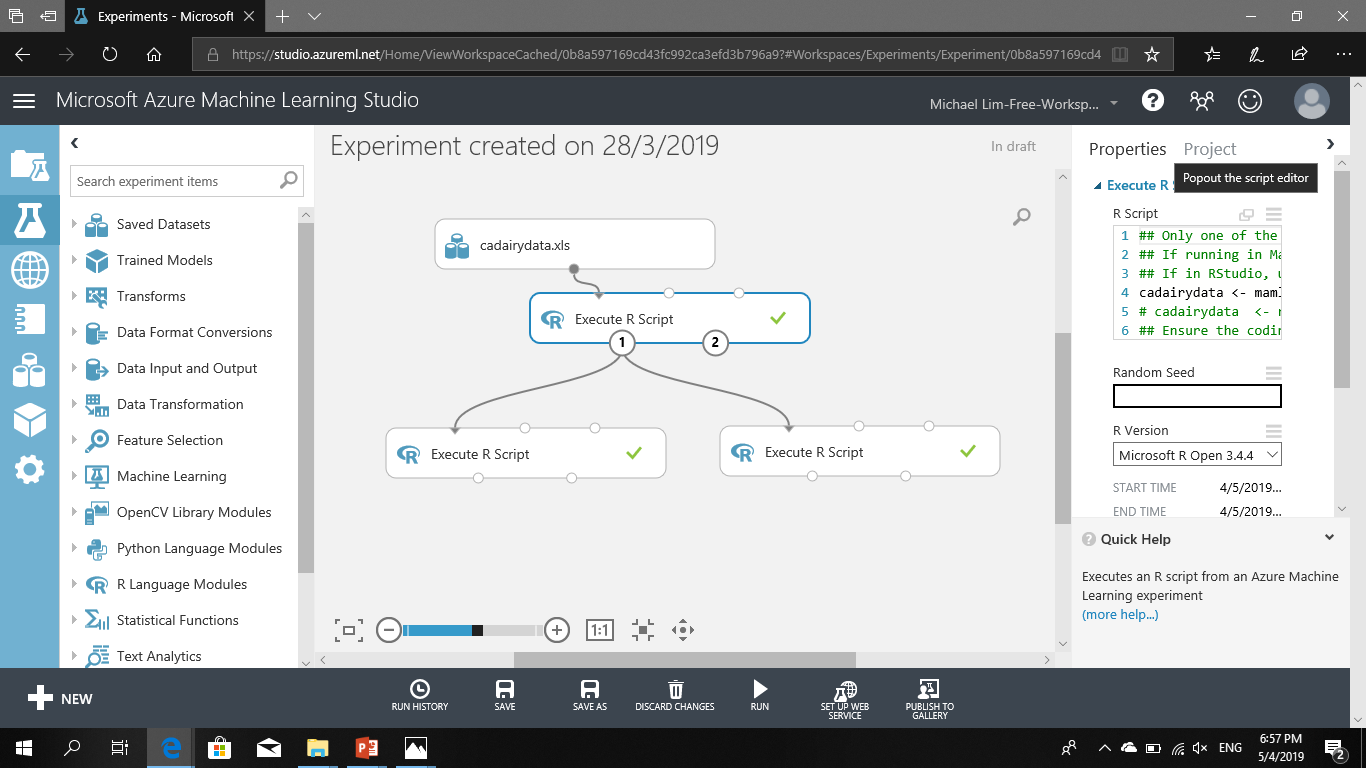
1. Setting up the Forecasting Model: (Explain the Process for setting up Forecasting Model using AML Studio)( screenshot of AML ) screenshot of all the above steps

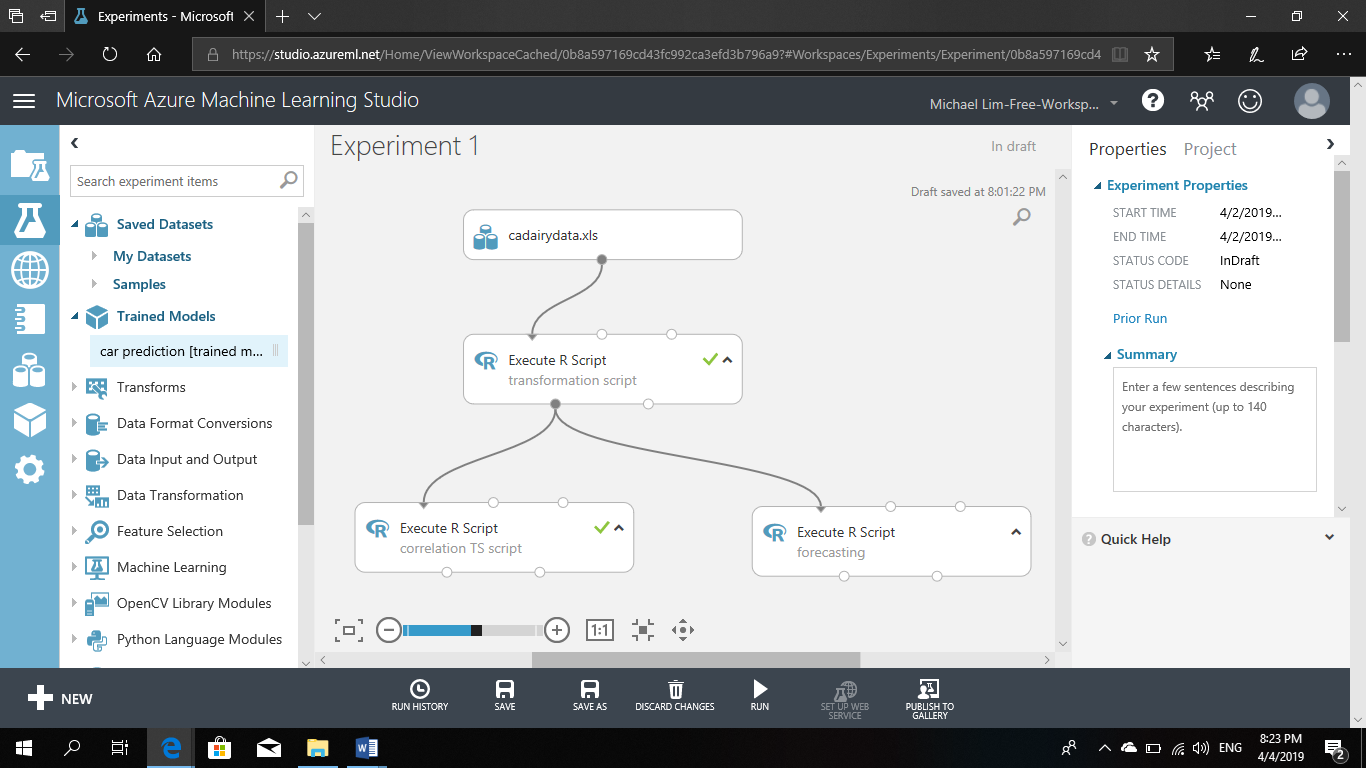












1. R Code for Data Filtering and Transformation: (Attach the R Code used in the Project for Data Transformation)( script1 ) -1,-2
   1. Filtering
      1. -1,-2

cadairydata <- cadairydata[ , c(-1,-2)]

* + 1. Factor , truncating

cadairydata$Month <- as.factor(substr(cadairydata$Month,1,3))

1. Transformed Data: (Attach the transformed data as Annexure)
   1. Transforming
      1. Monthcount

#3 create new column = month\_count

### function creation (arguments Year and Month)

month\_count <- function( Year, Month)

{

# define min\_year (statement)

min\_year <- min(Year)

# Compute the number of months from the start of the time series (object)

12 \* (Year - min\_year) + Month - 1

}

# compute the newly created function

cadairydata$Month.count <- month\_count(cadairydata$Year, cadairydata$Month.Number)

* + 1. [,4:7], fields transformed to same unit of measure

kpi\_uom <- function( kpi, uom = 1 )

{

log(kpi\*uom)

}

all\_uom <- list(1.0,6.5,1000.0,1000.0)

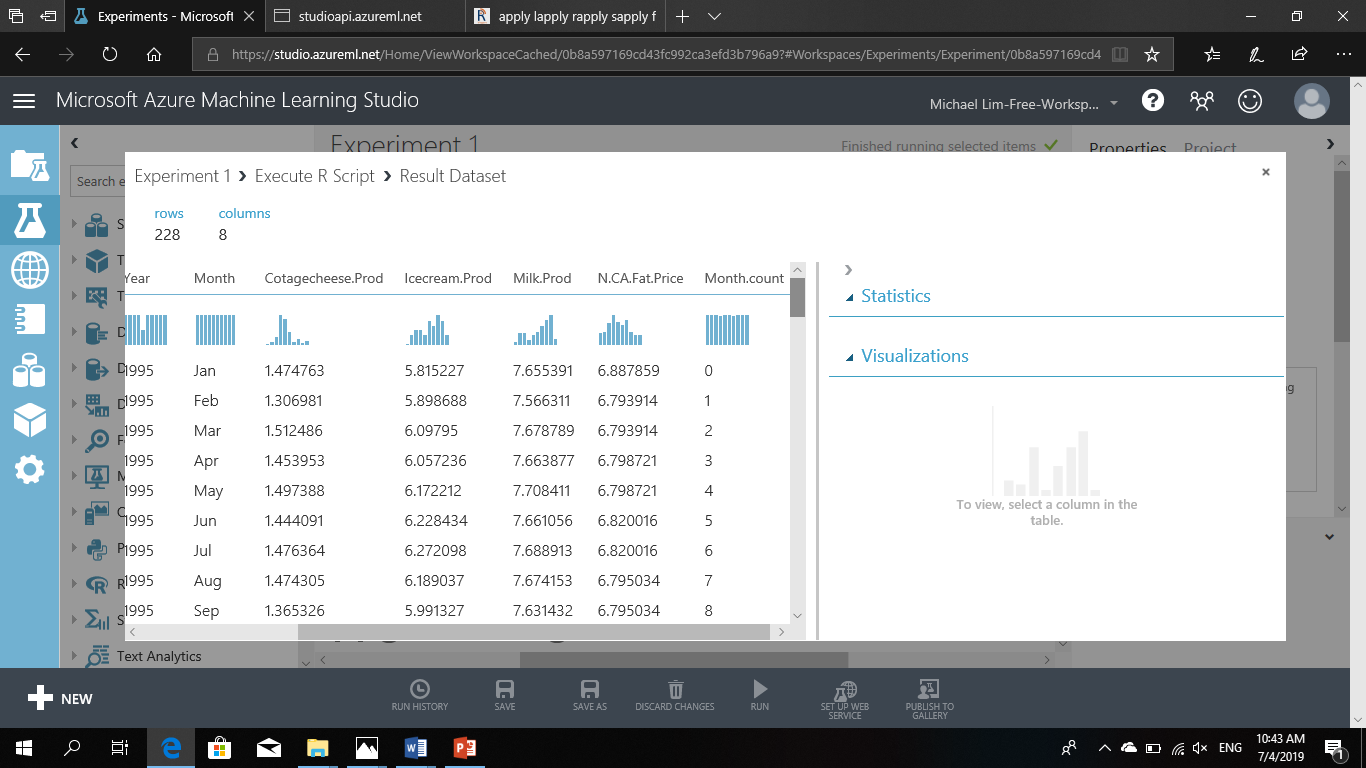
## function2 call

# old 4:7 values will be transformed to new values

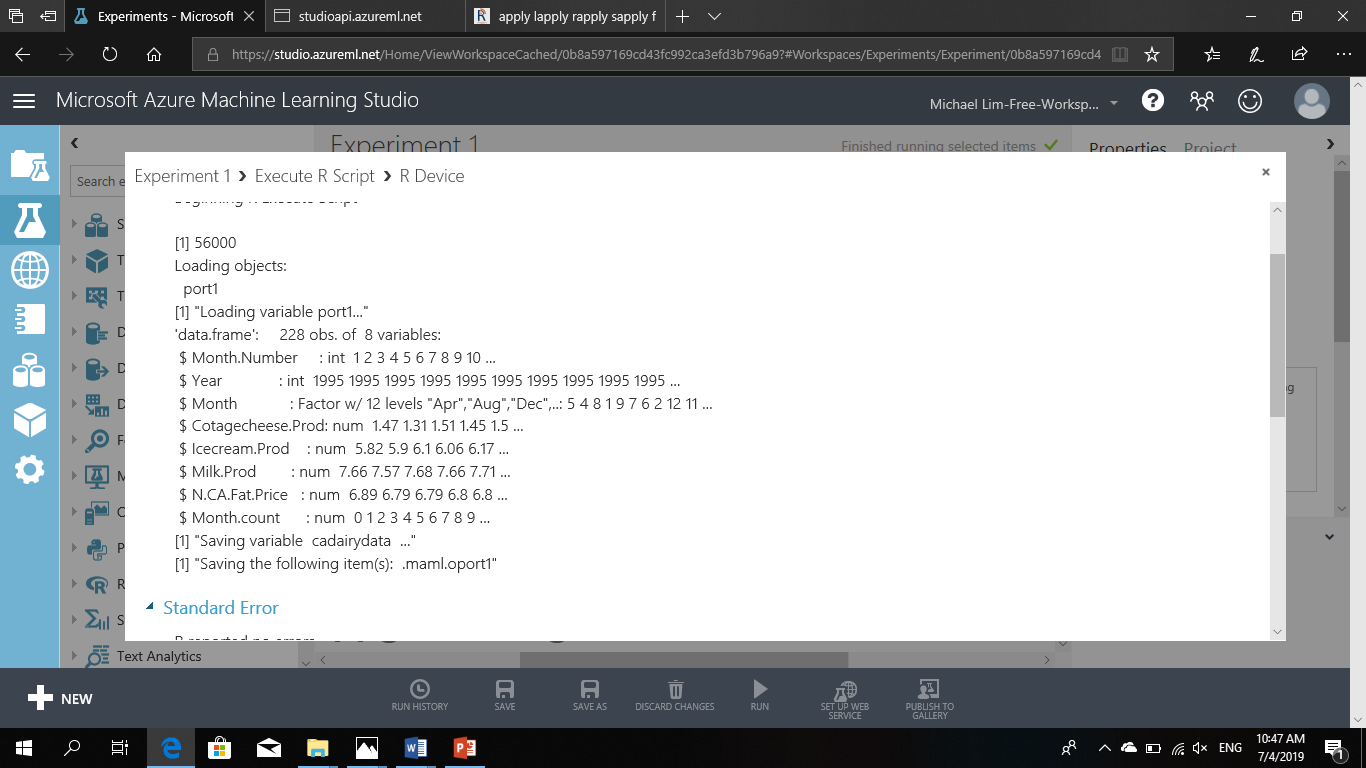
cadairydata[, 4:7] <- kpi\_uom( cadairydata[ ,4:7 ], all\_uom)

#cadairydata[, 4:7] <- Map(kpi\_uom , cadairydata[ ,4:7 ], all\_uom)

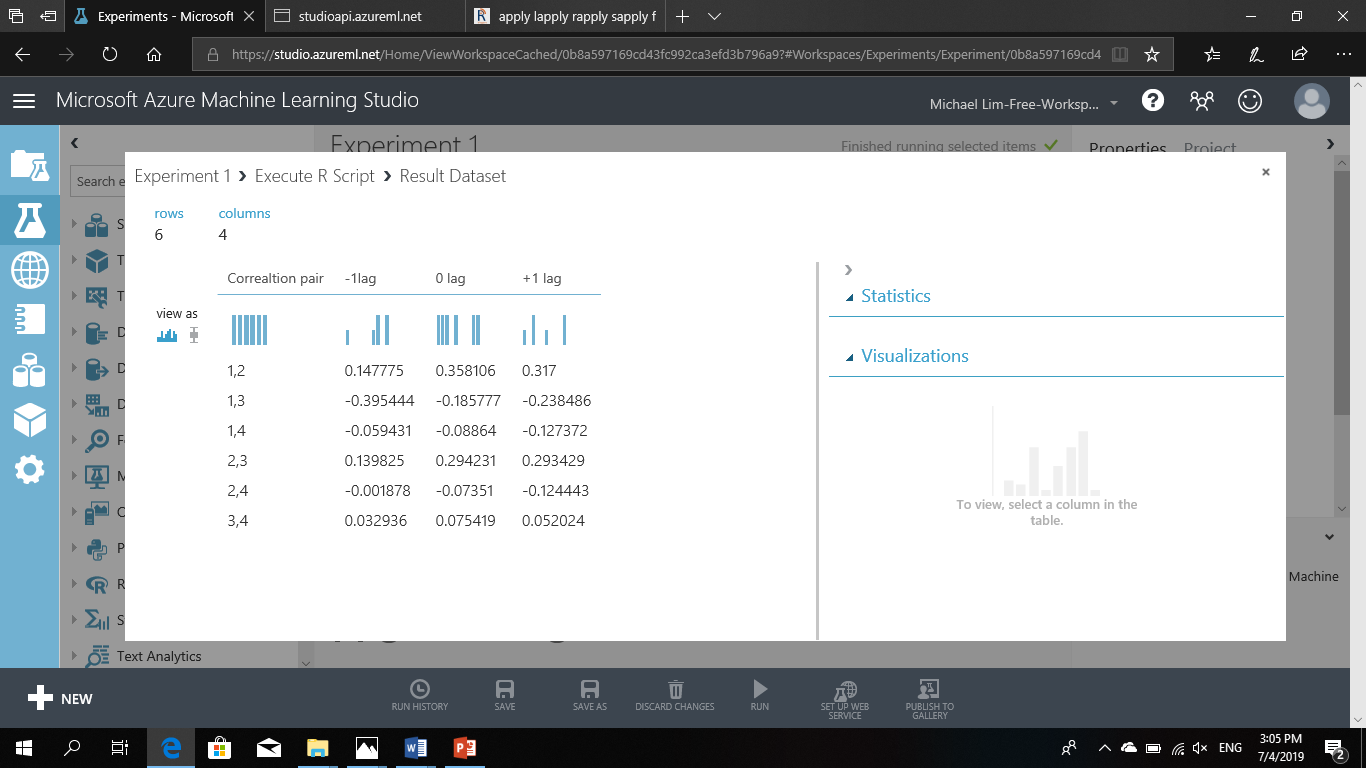
* + 1. Annexure A1( transformed data ) screenshot(
       1. MONTH must be 3 chars
       2. All [,4:7], must be same units of measure
       3. Month count



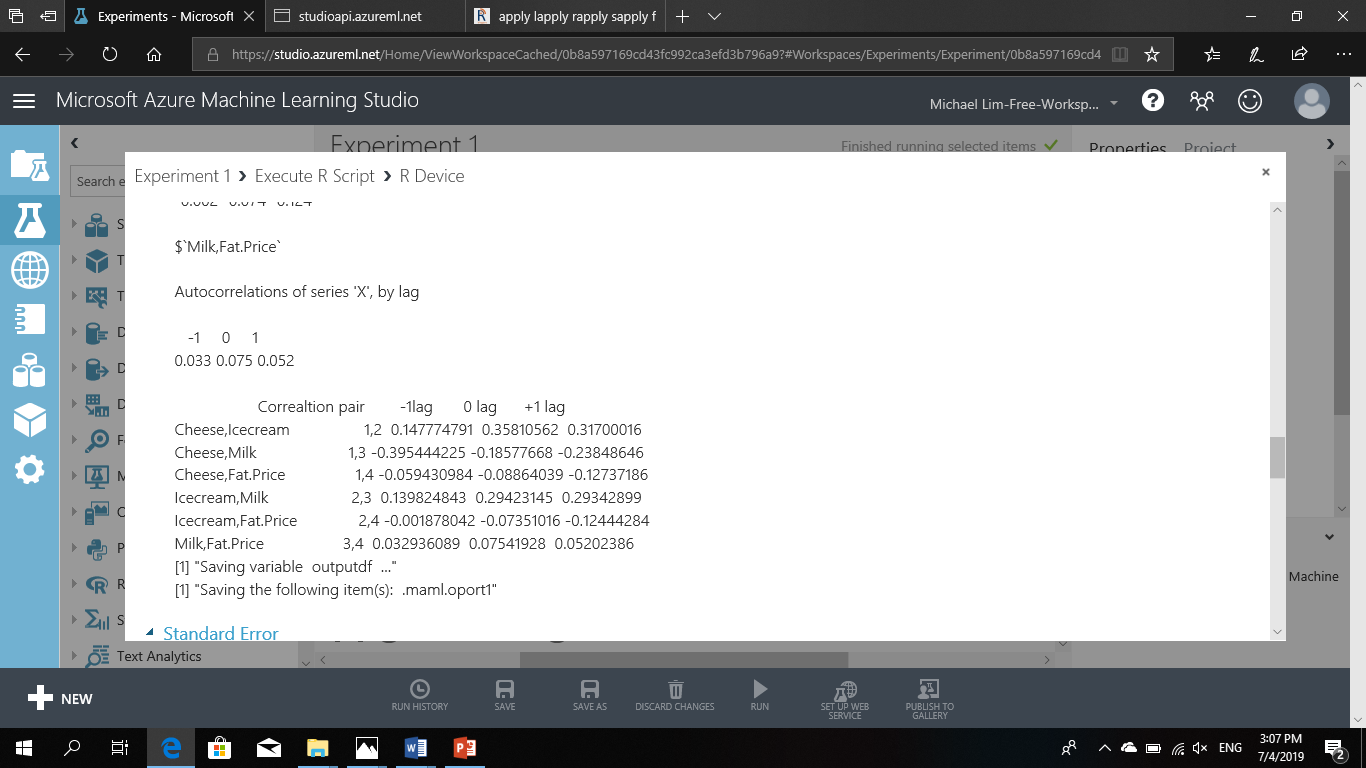
* + 1. Screenshot of rdevice visualize

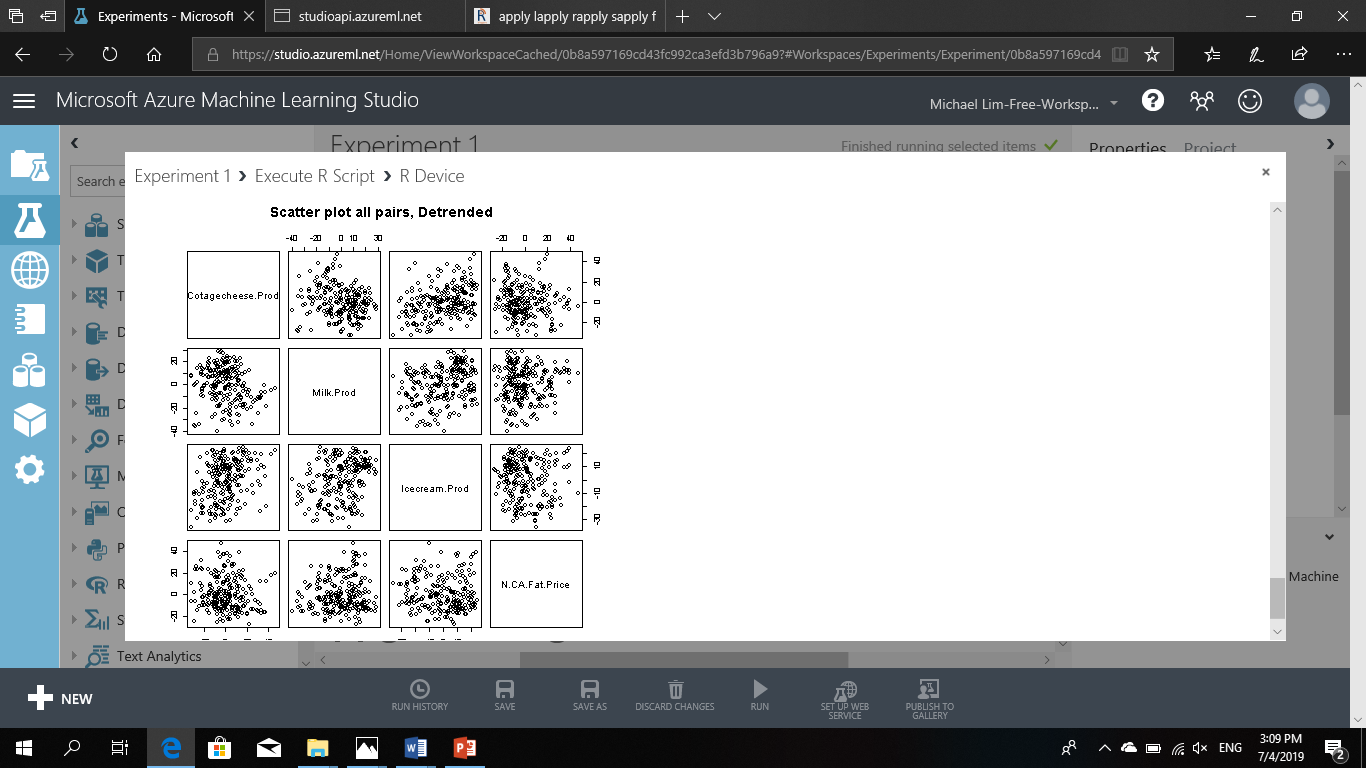


1. Correlation Analysis Process and Output: (Explain how you performed the Correlation analysis along with the output as Annexure)( script 2 output )
   1. Screenshot of pairs

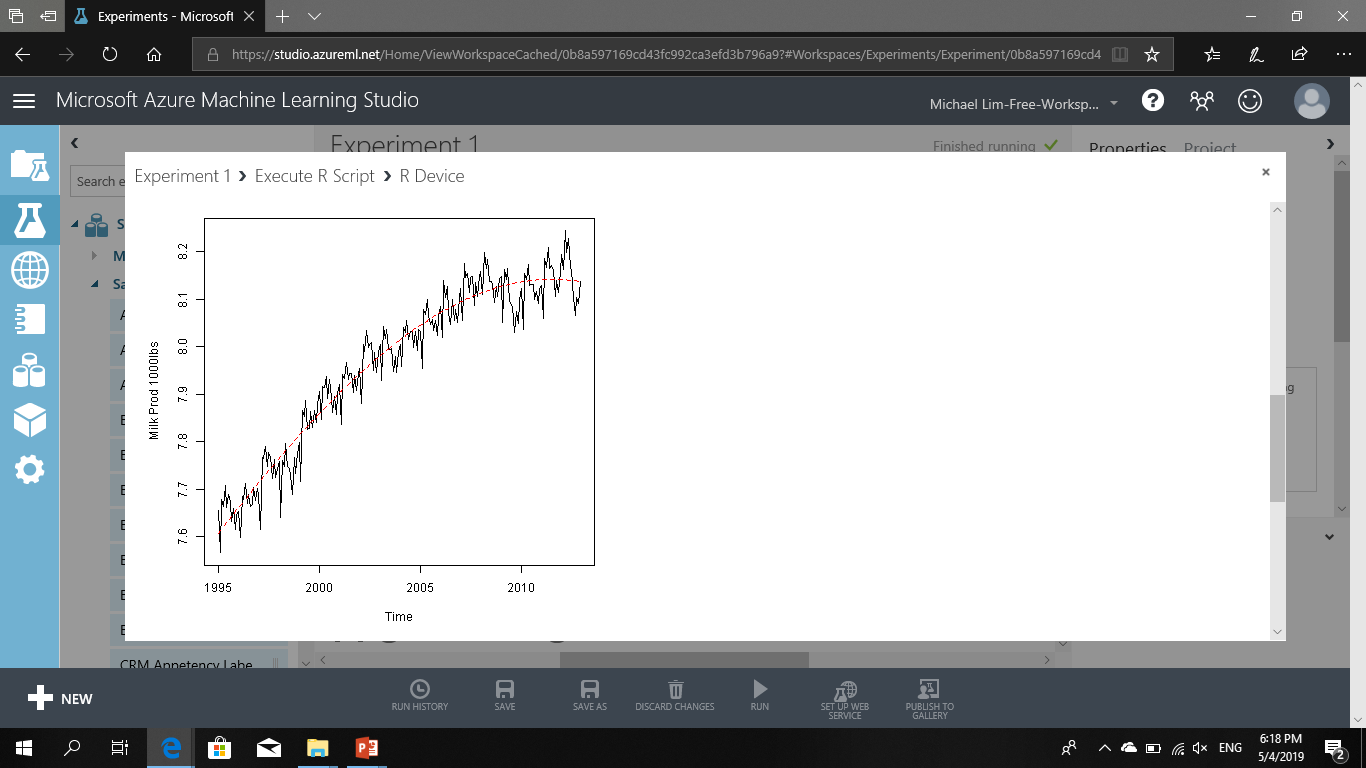


* 1. Output of script 2( full script 2)
     1. -1lag,0lag,+lag

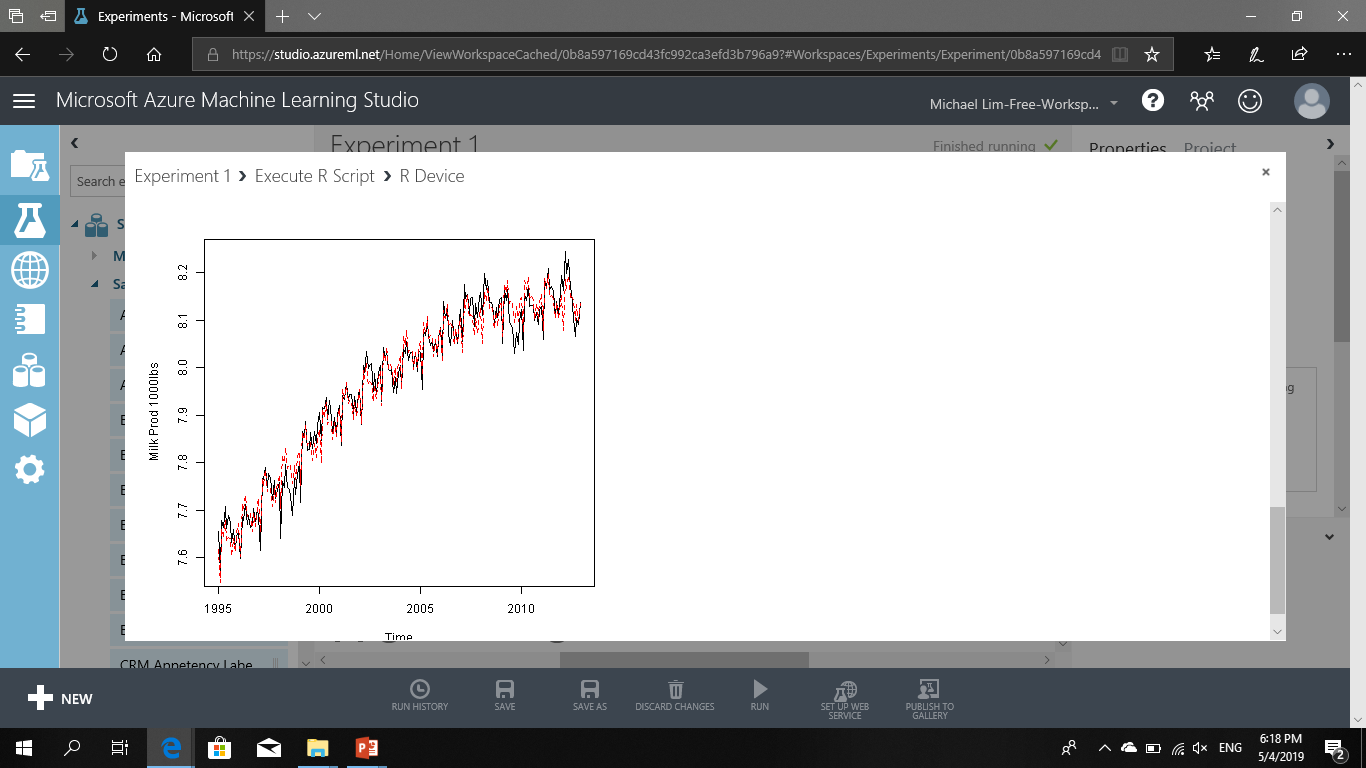


* 1. 

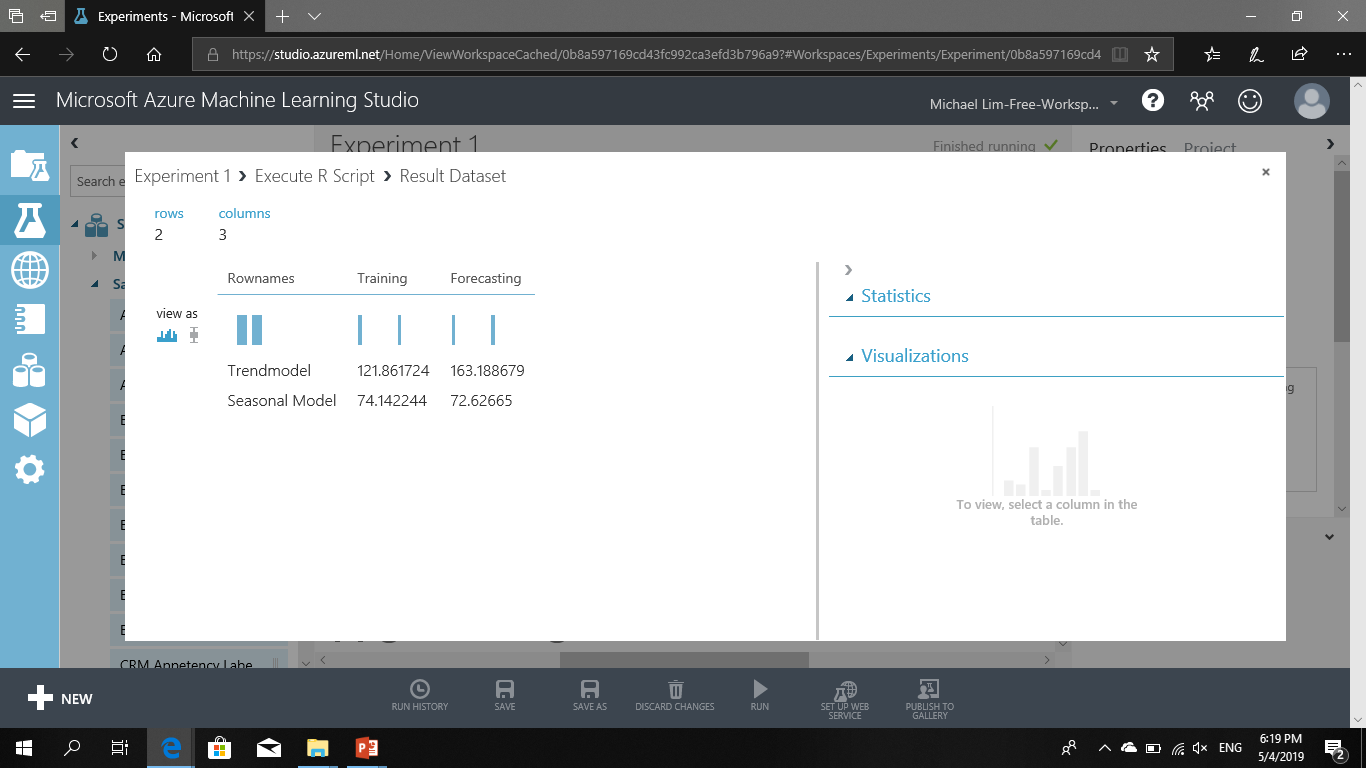
1. Implementing Seasonal Forecasting Model: (Explain how you have implemented the Seasonal Forecasting Model)script 3
   1. Trend model screenshot



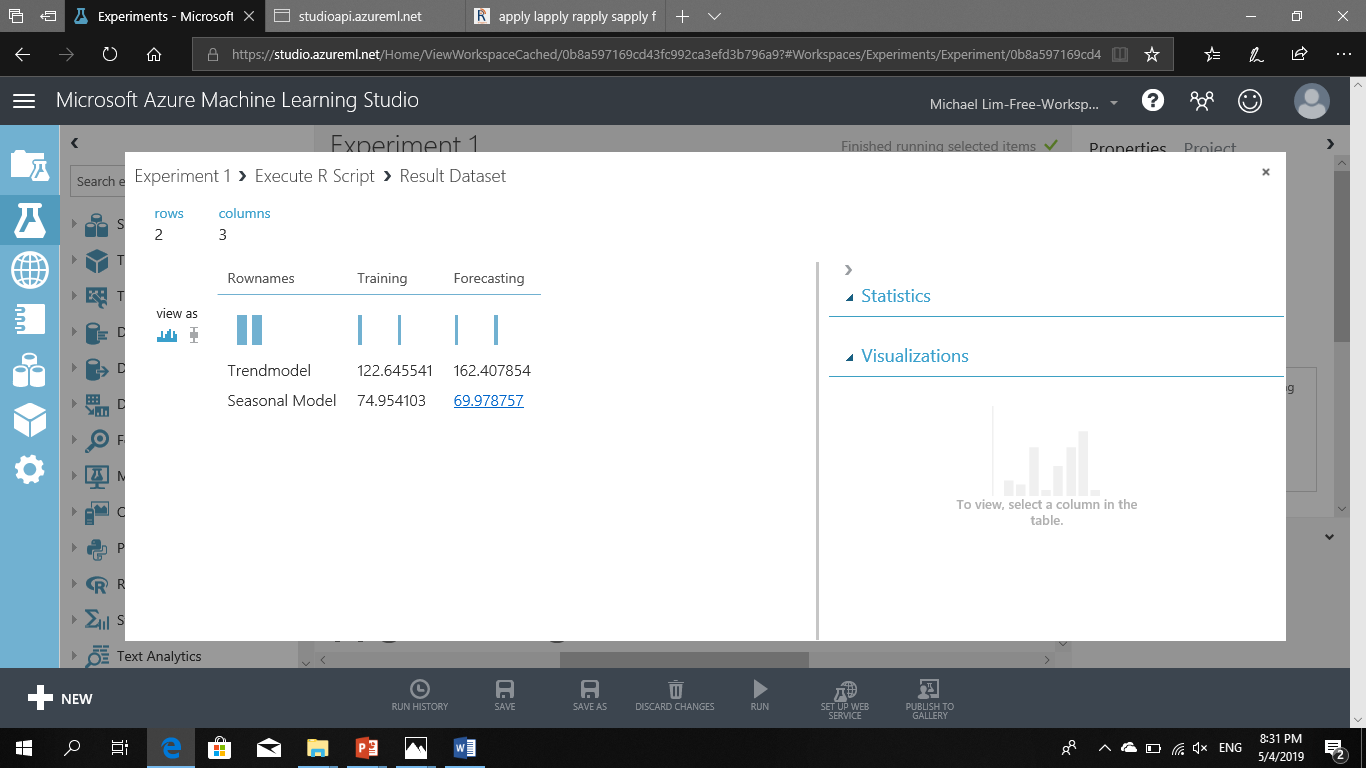
* 1. Forecasting model screenshot



* 1. RMS error ( comparsion of RMS root mean square error for trend and seasonal model ( outputport rms.df)



1. Refine Model: (Explain how you have refined the model along with Input, Process and Output)( script 3)

By changing the I(Month.count^2.5) in the linear model algorithm to I(Month.count^2) we get a lower RMS error score for both the seasonal and trend model. Overall we can see that the seasonal model also has a better fit with less error. 

1. Annexure 1
   1. Script1

# Map 1-based optional input ports to variables

cadairydata <- maml.mapInputPort(1) # class: data.frame

#1 remove first two columns

cadairydata <- cadairydata[ , c(-1,-2)]

# 2 categorzing month columns

cadairydata$Month <- as.factor(substr(cadairydata$Month,1,3))

#3 create new column = month\_count

### function creation (arguments Year and Month)

month\_count <- function( Year, Month)

{

# define min\_year (statement)

min\_year <- min(Year)

# Compute the number of months from the start of the time series (object)

12 \* (Year - min\_year) + Month - 1

}

# compute the newly created function

cadairydata$Month.count <- month\_count(cadairydata$Year, cadairydata$Month.Number)

# 4 Transformation of KPI to same unit of measurement (UOM)

## function 2 creation

kpi\_uom <- function( kpi, uom = 1 )

{

log(kpi\*uom)

}

all\_uom <- list(1.0,6.5,1000.0,1000.0)

## function2 call

# old 4:7 values will be transformed to new values

cadairydata[, 4:7] <- kpi\_uom( cadairydata[ ,4:7 ], all\_uom)

#cadairydata[, 4:7] <- Map(kpi\_uom , cadairydata[ ,4:7 ], all\_uom)

#5 remove missing values

cadairydata <- na.omit(cadairydata)

str(cadairydata)

# Select data.frame to be sent to the output Dataset port

maml.mapOutputPort("cadairydata");

1. Annexure 2
   1. Script 2

# Map 1-based optional input ports to variables

cadairydata <- maml.mapInputPort(1)

# class: data.frame

#Step 1 Creation of new column TIME as POSIXct object ( same as in Forecasting Script )

Sys.setenv(TZ = "PST8PDT")

# concatenating YEAR, MONTH NUMBER with yy,mm,dd,h,m,s, format

cadairydata$Time <- as.POSIXct(strptime(paste(as.character(cadairydata$Year), "-",

as.character(cadairydata$Month.Number),"-01 00:00:00", sep = ""),

"%Y -%m-%d %H :%M :%S"))

str(cadairydata)

# step 2

#PAIRS= correaltion between four columns , cotage chese, ice cream, milk production and Price KPI

# pairs(~ a + b + c + d ,

pairs( ~ Cotagecheese.Prod + Milk.Prod + Icecream.Prod + N.CA.Fat.Price , data = cadairydata,

main = "Scatter plot for 4 columns "

)

# creation of function for time series

detrend <- function( ts,Time )

{

# data frame creation

df\_time <- data.frame(ts = ts, Time = Time )

# remove linear modeling

ts <- ts - fitted(lm(ts ~ Time, data = df\_time ))

# calculte standrd deviation

ts.length <- length(ts)

stdev <- sqrt(sum((ts-mean(ts))^2))/(ts.length-1)

ts <- ts/stdev

ts

}

# data frame for all the four columns[ , 4:7] against TIME column using the above function

df.allcolumns <- data.frame(lapply(cadairydata[, 4:7], detrend, cadairydata$Time))

# pairs(~ a + b + c + d ,

pairs( ~ Cotagecheese.Prod + Milk.Prod + Icecream.Prod + N.CA.Fat.Price , data = cadairydata,

main = "Scatter plots for all pairs "

)

pairs( ~ Cotagecheese.Prod + Milk.Prod + Icecream.Prod + N.CA.Fat.Price , data = df.allcolumns ,

main = "Scatter plot all pairs, Detrended "

)

#5. Function to compute pairwise correlations list of 4 columns indexes

## function creation

pair.cor <- function(pair.ind, ts.list,lag.max = 1, plot = FALSE)

{

ccf(ts.list[[pair.ind[1]]], ts.list[[pair.ind[2]]], lag.max = lag.max, plot = plot)

}

## function call

## pairwise indexes in the form of a list

corpairs <- list(c(1,2), c(1,3), c(1,4),c(2,3), c(2,4),c(3,4))

cadairydatacorrelations <- lapply(corpairs,pair.cor,df.allcolumns)

names(cadairydatacorrelations) <- c("Cheese,Icecream","Cheese,Milk","Cheese,Fat.Price","Icecream,Milk","Icecream,Fat.Price","Milk,Fat.Price")

str(cadairydatacorrelations)

cadairydatacorrelations

df.correlations <- data.frame(do.call(rbind, lapply(cadairydatacorrelations,'[[',1)))

#("1,2","1,3","1,4","2,3","2,4","3,4") - but with names and spacings for readability

c.names <- c("Correaltion pair", "-1lag","0 lag", "+1 lag")

r.names <- c("1,2","1,3","1,4","2,3","2,4","3,4")

outputdf <- cbind(r.names, df.correlations)

colnames(outputdf) <- c.names

outputdf

# Select data.frame to be sent to the output Dataset port

maml.mapOutputPort("outputdf");

1. Annexure 3
   1. Script 3

# Map 1-based optional input ports to variables

cadairydata <- maml.mapInputPort(1) # class: data.frame

#Step 1 Creation of new column TIME as POSIXct object ( same as in Script 2 )

Sys.setenv(TZ = "PST8PDT")

cadairydata$Time <- as.POSIXct(strptime(paste(as.character(cadairydata$Year), "-",

as.character(cadairydata$Month.Number),"-01 00:00:00", sep = ""),

"%Y -%m-%d %H :%M :%S"))

str(cadairydata)

# step 2 Creation of training Dataset

dairy\_train <- cadairydata[ 1: 216, ] # all columns 216 records for training ( TRAIN )

# Step 3 Trend Model

### 3.1 linear model algorithm

milk.tm <- lm(Milk.Prod~Time + I(Month.count^2), data = dairy\_train)

### 3.2 plot for time and milk production from the training dataset

plot(dairy\_train$Time, dairy\_train$Milk.Prod, xlab = "Time", ylab = "Milk Prod 1000lbs", type = 'l')

###3.3 lines of training dataset's Time which can be precited from the linear model ( milk.tm)

lines(dairy\_train$Time, predict(milk.tm, dairy\_train), lty = 2, col = 2)

b. # Step 4 seasonal Model

### 4.1 linear model algorithm

milk.sm <- lm(Milk.Prod~Time + I(Month.count^2) + Month - 1 , data = dairy\_train)

### 4.2 plot for time and milk production from the training dataset

plot(dairy\_train$Time, dairy\_train$Milk.Prod, xlab = "Time", ylab = "Milk Prod 1000lbs", type = 'l')

###4.3 lines of training dataset's Time which can be precited from the linear model ( milk.tm)

lines(dairy\_train$Time, predict(milk.sm, dairy\_train), lty = 2, col = 2)

# Step 5 Predictin

## 5.1 Predict trend model

predict\_tm <- predict(milk.tm, cadairydata)

## 5.2 Predict Seasonal model

predict\_sm <- predict(milk.sm, cadairydata)

### 6 RMSE

## 6.1 Create function

rmserror <- function( series1, series2)

{

temp1 <- exp(series1)

temp2 <- exp(series2)

sqrt(sum((temp1-temp2)^2)/ length(temp1))

}

## 6.2 create data frame

######## ROW NAMES with TREND MODEL AND SEASONAL MODEL

######## COLUMN NAME WITH TRAINING AND FORECASTING

rms.df <- data.frame(

Rownames = c("Trendmodel", "Seasonal Model "),

Training = c(rmserror(predict\_tm[1:216],cadairydata$Milk.Prod[1:216] ),

rmserror(predict\_sm[1:216],cadairydata$Milk.Prod[1:216] )),

Forecasting = c(rmserror(predict\_tm[217:228],cadairydata$Milk.Prod[217:228] ),

rmserror(predict\_sm[217:228],cadairydata$Milk.Prod[217:228] ))

)

rms.df

# Select data.frame to be sent to the output Dataset port

maml.mapOutputPort("rms.df");

1. Forecast Output after Model Adjustment: (Explain the Forecast Output after you have adjusted the model based on actual production data)( output of script 3 )

The RMSE is lower for the seasonal model than the trend model, owing to the fact that milk production is seasonal. Also, the changing the value of 2.5 to 2 in I(Month.count^2.5) changes the n value in polynomial regression and changes the fit of the line. It must be adjusted through trial and error although large numbers should be avoided to prevent “picking up the noise” or “overfitting”

1. conclusions

We have learnt through this Project how to use R programming in conjunction with Azure Machine Learning Studio, check data for strong or weak correlations and to create Trend and Seasonal models and how to assess which models are more accurate.