Dublin Business School

B9DA101 Statistics for Data Analytics: CA_TWO

January 2025

Submitted by:
Anish Rao: 20066423

Lecturer: Dr. Muhammad Alli

Index

Introduction

Purpose of the Analysis

This analysis explores the application of statistical techniques across three different problem domains: regression modeling, Bayesian estimation, and time series forecasting.

The report is divided into three main questions:

- **Question 1:** Applying a Generalized Linear Model (GLM).
- **Question 2:** Constructing a Bayesian estimation framework with Poisson-distributed data.
- **Question 3:** Analyzing a financial time series and forecasting future trends.

Dataset Overview

Three datasets were used in this analysis:

- **mtcars (Q1):**
  - A built-in dataset in R with 32 observations on fuel consumption and 10 design aspects of cars.
  - Used to model transmission type (am) based on variables like mpg, hp, wt, and cyl.
- **Simulated Poisson Data (Q2):**
  - A sample of 10 values generated from a Poisson(λ=3) distribution.
  - Used to demonstrate likelihood computation and Bayesian inference using a Gamma prior.
- **EuStockMarkets (Q3):**
  - Time series dataset of daily closing prices of four major European stock indices.
  - The DAX index was used to perform stationarity checks, ARIMA modeling, and forecasting.

Google Colab Notebook link-

https://colab.research.google.com/drive/18wMC0CFy5MYsx3ItcEJew90EmbEyLf7C?usp=sharing

Q1. Consider a relational dataset and specify your input and output variables

The dataset used is **`mtcars`** which is inbuilt in RStudio.

Input variables - **`mpg`**, **`hp`**, **`wt`**, **`cyl`**

Output variable - **`am`**

```
data("mtcars")
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

(a) Logistic model trained using 80% of the dataset

Train the model using 80% of this dataset and suggest an appropriate GLM to model output to input variables. (10)

Code -

```
mtcars$am = as.factor(mtcars$am)
mtcars$cyl = as.factor(mtcars$cyl)

data_split = createDataPartition(mtcars$am, p = 0.8, list = FALSE)
train_data = mtcars[data_split, ]
test_data = mtcars[-data_split, ]

model = glm(am ~ mpg + hp + wt + cyl, data = train_data, family = binomial)
```

The variables `am` and `cyl` are converted to factors. The dataset is split into training (80%) and testing (20%) sets. A Binomial regression model is then trained to predict the transmission type (`am`).

(b) Significant variables and estimated parameters

Specify the significant variables on the output variable at the level of α=0.05 and explore the related hypotheses test. Estimate the parameters of your model. (10)

Code –

```
summary(model)
```

Output –

```
## Call:
## glm(formula = am ~ mpg + hp + wt + cyl, family = binomial, data = train_data)
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1280.950 809723.749  -0.002    0.999
## mpg              40.998  28196.890   0.001    0.999
## hp                6.413   3310.121   0.002    0.998
## wt              -98.671  58606.465  -0.002    0.999
## cyl6             -5.707  30697.754   0.000    1.000
## cyl8           -480.280 345022.731  -0.001    0.999
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.6499e+01  on 26  degrees of freedom
## Residual deviance: 7.3221e-09  on 21  degrees of freedom
## AIC: 12
##
## Number of Fisher Scoring iterations: 25
```

Insights -

At the α = 0.05 level, none of the variables are statistically significant as all p-values are greater than 0.05.

Hypotheses:

- Null hypothesis (H_0): The variable has no effect on the output (**`am`**)
- Alternative hypothesis (H_1): The variable does affect the output (**`am`**)

As all p- values > 0.05, we fail to reject the null hypothesis for all variables.

Estimated parameters:

Variable	Coefficient
Intercept	-1280.950
mpg	40.998
hp	6.413
wt	-98.671
cyl6	-5.707
cyl8	-480.280

(c) Predict outcomes for test data and model equation

Predict the output of the test dataset using the trained model. Provide the functional form of the optimal predictive model. (10)

Code -

```
predicted_prob = predict(model, newdata = test_data, type = "response")

predicted_class = ifelse(predicted_prob > 0.5, 1, 0)
actual_class = test_data$am
```

The model predicts probabilities on the test set and converts them into class labels using a 0.5 threshold.

Output -

```
print(data.frame(Actual = actual_class, Predicted = predicted_class))

##                   Actual Predicted
## Mazda RX4 Wag          1         0
## Duster 360             0         1
## Merc 280C              0         0
## Dodge Challenger       0         0
## Porsche 914-2          1         1
```

Insights -

The model output returns the predicted transmission class (`0` or `1`) for each test data point.
Functional Form:

$$\log\left(\frac{p}{1-p}\right) = -1280.95 + 40.998 \cdot mpg + 6.413 \cdot hp - 98.671 \cdot wt - 5.707 \cdot cyl_6 - 480.280 \cdot cyl_8$$

Here, p = probability that the car has a manual transmission (**`am`** = 1).

(d) Confusion matrix and accuracy

Provide the confusion matrix and obtain the probability of correctness of predictions. (5)

Code -

```
predicted_class = as.factor(predicted_class)
actual_class = as.factor(actual_class)

confusion_matrix = confusionMatrix(predicted_class, actual_class)
```

A confusion matrix is created to compare predicted and actual classes for the test data.

Output -

```
print(confusion_matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 2 1
##          1 1 1
##                Accuracy : 0.6
##                  95% CI : (0.1466, 0.9473)
##     No Information Rate : 0.6
##     P-Value [Acc > NIR] : 0.6826
##
##                   Kappa : 0.1667
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.6667
##             Specificity : 0.5000
##          Pos Pred Value : 0.6667
##          Neg Pred Value : 0.5000
##              Prevalence : 0.6000
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.6000
##       Balanced Accuracy : 0.5833
##
##        'Positive' Class : 0
```

Insights -

The model got 3 correct out of 5 and has **60%** accuracy.

Q2. Let x_1,…,x_10 are identically independently distributed (iid) with Poisson(λ)

Generated 10 independent values from a Poisson distribution with λ = 3.

```
x = rpois(10, lambda = 3)
print(x)

##  [1] 4 4 2 1 4 5 1 3 3 3
```

(a) Likelihood function for a given λ

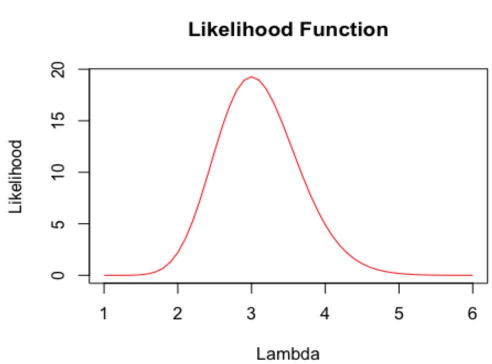
Compute the likelihood function (LF). (10)

Code -

```
likelihood = function(lambda, data) {
  n = length(data)
  sum_x = sum(data)
  return((lambda ^ sum_x) * exp(-n * lambda))
}

lambda_values = seq(1, 6, by = 0.1)
likelihood_values = sapply(lambda_values, likelihood, data = x)

plot(lambda_values, likelihood_values,
     type = "l",
     col = "red",
     main = "Likelihood Function",
     xlab = "Lambda", ylab = "Likelihood")
```

The function calculates the likelihood of observing the data for different λ values.

Output -



Insights -

The likelihood function shows how likely different values of λ are, given the observed data. In this case, the likelihood peaks around λ = 3.

(b) Gamma prior for λ using chosen hyperparameters

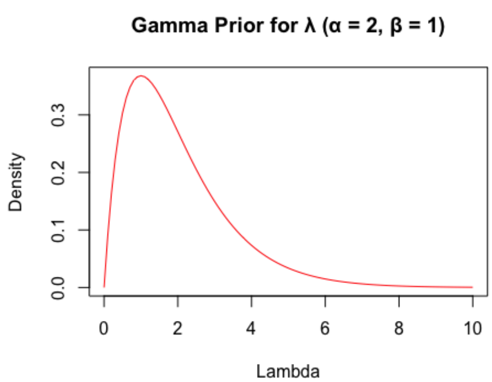
Adopt the appropriate conjugate prior to the parameter λ (Hint: Choose hyperparameters optionally within the support of distribution). (10)

Code -

```
alpha = 2
beta = 1
lambda_vals = seq(0, 10, by = 0.1)

prior_density = dgamma(lambda_vals, shape = alpha, rate = beta)

plot(lambda_vals, prior_density,
     type = "l",
     col = "red",
     main = "Gamma Prior for λ (α = 2, β = 1)",
     xlab = "Lambda", ylab = "Density")
```

A Gamma distribution with shape $\alpha = 2$ and rate $\beta = 1$ is used to represent prior for λ.

Output -



Insights -

The Gamma distribution is a natural conjugate prior for Poisson(λ).

(c) Posterior distribution of λ

Using (a) and (b), find the posterior distribution of λ. (10)

Code -

```
sum_x = sum(x)
n = length(x)

posterior_alpha = alpha + sum_x
posterior_beta = beta + n
```

The posterior parameters are calculated by updating the prior with the observed data: $\alpha = \alpha + \text{sum}(x)$, and $\beta = \beta + n$.

Output -

```
cat("Posterior: \nGamma (",posterior_alpha,",",posterior_beta,")\n")

## Posterior:
## Gamma ( 32 , 11 )
```

Insights -

Based on the observed data and the prior (Gamma(2, 1)), the posterior distribution for λ is Gamma(32, 11)

(d) Bayesian estimator of λ

Compute the minimum Bayesian risk estimator of λ. (5)

Code -

```
lambda_bayes = posterior_alpha / posterior_beta
```

The posterior mean (α / β) is used as the Bayesian estimator of λ.

Output -

```
cat("Bayesian Estimator:", lambda_bayes)

## Bayesian Estimator: 2.909091
```

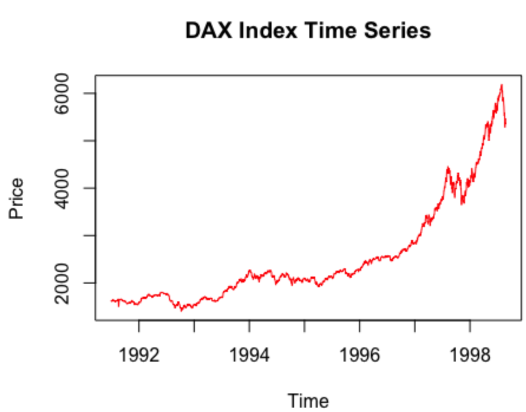
Insights -

The Bayesian estimator represents the expected value of λ based on the posterior distribution.
For Gamma(32, 11), this gives an estimate ~ 2.91.

Q3. Use a particular stock market dataset to accomplish the time series analysis

The dataset used is `EuStockMarkets` which is inbuilt in RStudio.

```
data("EuStockMarkets")
head(EuStockMarkets)

## Time Series:
## Start = c(1991, 130)
## End = c(1991, 135)
## Frequency = 260
##               DAX    SMI    CAC   FTSE
## 1991.496 1628.75 1678.1 1772.8 2443.6
## 1991.500 1613.63 1688.5 1750.5 2460.2
## 1991.504 1606.51 1678.6 1718.0 2448.2
## 1991.508 1621.04 1684.1 1708.1 2470.4
## 1991.512 1618.16 1686.6 1723.1 2484.7
## 1991.515 1610.61 1671.6 1714.3 2466.8

str(EuStockMarkets)

##  Time-Series [1:1860, 1:4] from 1991 to 1999: 1629 1614 1607 1621 1618 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "DAX" "SMI" "CAC" "FTSE"

summary(EuStockMarkets)

##       DAX            SMI            CAC            FTSE
##  Min.   :1402   Min.   :1587   Min.   :1611   Min.   :2281
##  1st Qu.:1744   1st Qu.:2166   1st Qu.:1875   1st Qu.:2843
##  Median :2141   Median :2796   Median :1992   Median :3247
##  Mean   :2531   Mean   :3376   Mean   :2228   Mean   :3566
##  3rd Qu.:2722   3rd Qu.:3812   3rd Qu.:2274   3rd Qu.:3994
##  Max.   :6186   Max.   :8412   Max.   :4388   Max.   :6179

dax = EuStockMarkets[, "DAX"]
plot(dax,
     type = "l",
     col = "red",
     main = "DAX Index Time Series",
     xlab = "Time", ylab = "Price")
```



(a) Check if mean and variance are stationary

Check whether the time series is stationary in mean and variance. (5)

Code -

```
adf_test = adf.test(dax)
print(adf_test)

sd_full = sd(dax)
sd_first_half = sd(dax[1:(length(dax)/2)])
sd_second_half = sd(dax[((length(dax)/2) + 1):length(dax)])

cat("Full SD:", sd_full, "\n")

cat("First Half SD:", sd_first_half, "\n")

cat("Second Half SD:", sd_second_half, "\n")
```

The `adf.test()` checks for stationarity in mean, while standard deviations for the full series and each half are calculated to check difference in variance.

Output -

```
##  Augmented Dickey-Fuller Test
##
## data:  dax
## Dickey-Fuller = -0.82073, Lag order = 12, p-value = 0.9598
## alternative hypothesis: stationary

## Full SD: 1084.793

## First Half SD: 241.1517

## Second Half SD: 1137.395
```

Insights -

Hypotheses:

- Null hypothesis (H_0): Series mean is non-stationary.
- Alternative hypothesis (H_1): Series mean is stationary.

As p- value (0.96) > 0.05, we fail to reject the null hypothesis , therefore the series is not stationary in mean.

Standard Deviation also changes a lot between the first and second half, therefore the series is not stationary in variance.

(b) Plot acf() and pacf()

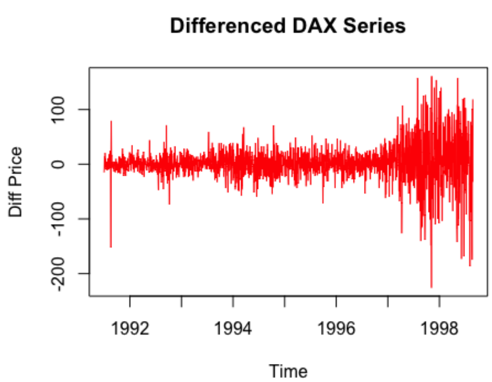
Use acf() and pacf() functions to identify the order of AR and MA. (10)

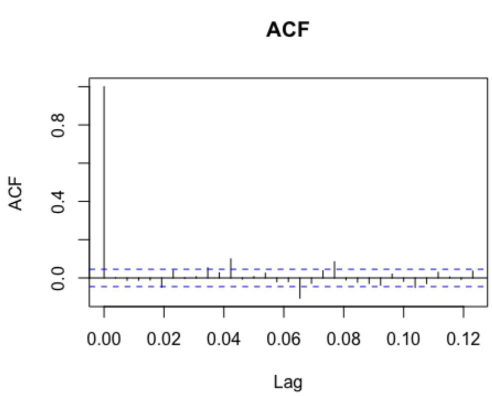
Code -

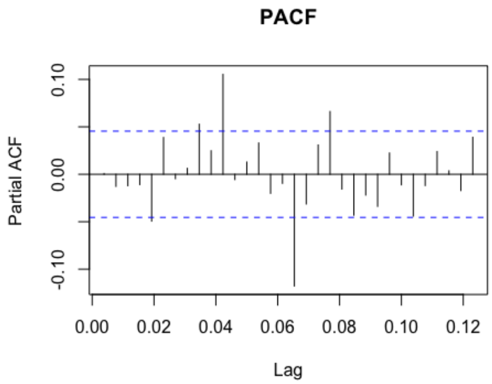
```
dax_diff = diff(dax)

plot(dax_diff,
     type = "l",
     col = "red",
     main = "Differenced DAX Series",
     xlab = "Time", ylab = "Diff Price")

acf(dax_diff, main = "ACF")

pacf(dax_diff, main = "PACF")
```

The series is first differenced using `diff()` to stabilize the mean. The `acf()` and `pacf()` plots are then used to examine autocorrelation and partial autocorrelation patterns.

Output -







Insights -

The ACF drops sharply after lag 1 and PACF shows a few significant spikes.

(c) Best ARIMA model

Use auto.arima() to learn the best ARIMA model. (5)

Code -

```
best_model = auto.arima(dax)
print(best_model)
```

The `auto.arima()` function automatically selects the best-fitting ARIMA model for the DAX time series based on information criteria like AIC and BIC.

Output -

```
## Series: dax
## ARIMA(5,2,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5
##       -0.8187  -0.6631  -0.5053  -0.3444  -0.2231
## s.e.   0.0228   0.0287   0.0306   0.0289   0.0231
##
## sigma^2 = 1235:  log likelihood = -9248
## AIC=18507.99   AICc=18508.04   BIC=18541.15
```

Insights -

The chosen model is ARIMA(5,2,0), which means the series was differenced twice to achieve stationarity and includes five autoregressive terms.

(d) 10 step Forecast

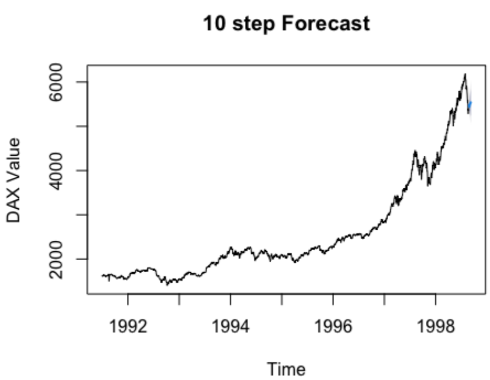
Forecast h=10 step ahead prediction of the time series variable and plot it with the original time series. (10)

Code -

```
forecast_result = forecast(best_model, h = 10)

plot(forecast_result,
     main = "10 step Forecast",
     xlab = "Time", ylab = "DAX Value")
```

The `forecast()` function generates a 10-step ahead prediction using the selected ARIMA model. The results are plotted to visualize future values alongside the original time series.

Output -



Insights -

The forecast shows a continued upward trend in DAX values. The confidence intervals widen over time, indicating increasing uncertainty in long-term predictions.