USA AIRLINE DELAYS AND THEIR IMPACTS

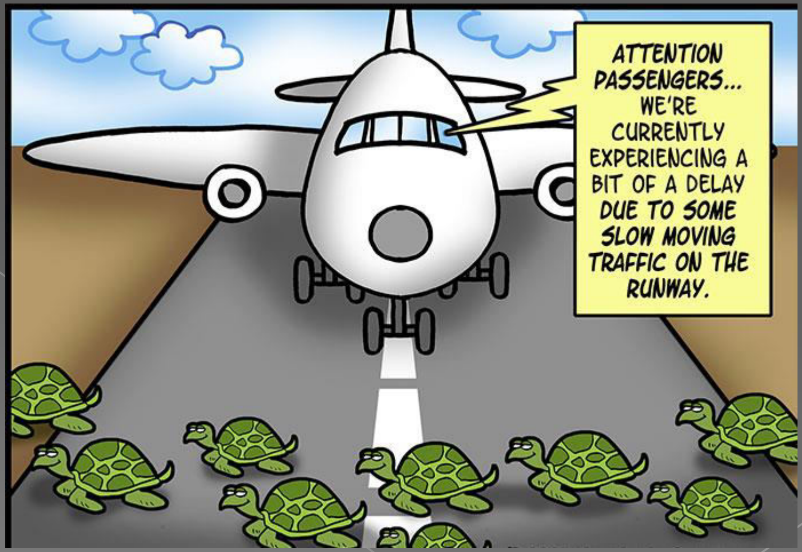
Explained using Multiple Regression and Logistic Regression

-Satya M

Why Airline Delays?

Cost of Delays in the US

$32.9 B			
$8.3 B	$16.7 B	$3.9 B	$4 B
Cost to Airlines	Cost to Passengers	Cost from Lost Demand	GDP Impact



How Different from Existing Models?

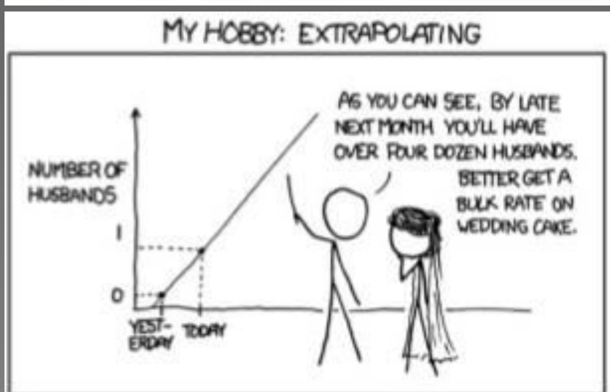
- I found 'N' number of projects in the internet trying to predict the Air Delay time based on various factors.
- However, I would focus on two important factors.
  - AirTime
  - WeatherDelay
- I'm using the above factors for consideration as the same hasn't been used before in any model and they are quite difficult to predict.
- They play an significant factor while looking at Air Line information.

About the Dataset

- I used the dataset from Kaggle and it was populated using "The Flight Delay" project which is the most complete database of flight delays in the United States
- The dataset is separated into individual documents – one record for every year – and contains around 7 million records each.
- I'm considering the records only for the year 2008, as it holds the most recent data. Further, I'm looking at flight data with of Chicago and Washington DC.
- The source of data is given in the below link
- https://www.kaggle.com/giovamata/airlinedelaycauses

Carrier Delay
UA 628 1,637
Flight Num: 628
Unique Carrier: UA
Carrier Delay: 1,637
UA 624 1,132
UA 626 1,132
UA 620 922
AA 1544 1,067
AA 1442 897
UA 616 802
UA 612 700
UA 602 520
UA 610 489
AA 810 765
AA 580 598
UA 614 729
UA 604 678
UA 606 469
UA 608 406
AA 1448 512
AA 2328 376
AA 496
UA 618 706
UA 622 617
UA 600 301
UA 1166 236
MQ 3908 1,045
Carrier Delay
215
1,637
Weather Delay
AA
UA
Unique Carrier
AA
MQ
UA
NAS Delay
Unique Carrier
NAS Delay
6K
5K
4K
3K
2K
1K
0K
AA
MQ
UA
TaxiIn and TaxiOut vs Carrier
Unique Carrier
AA
MQ
UA
0
500
1000
1500
2000
2500
Taxi In
0K
2K
4K
6K
8K
10K
12K
14K
Taxi Out

What is Regression?



❖ Regression Analysis is the art and science of fitting straight lines to patterns of data.

❖ Regression Analysis is widely used for predicting and forecasting.

❖ In a linear regression model, the variable of interest which is the dependent variable is predicted from a single (simple linear regression) or multiple group of independent variables (multiple linear regression).

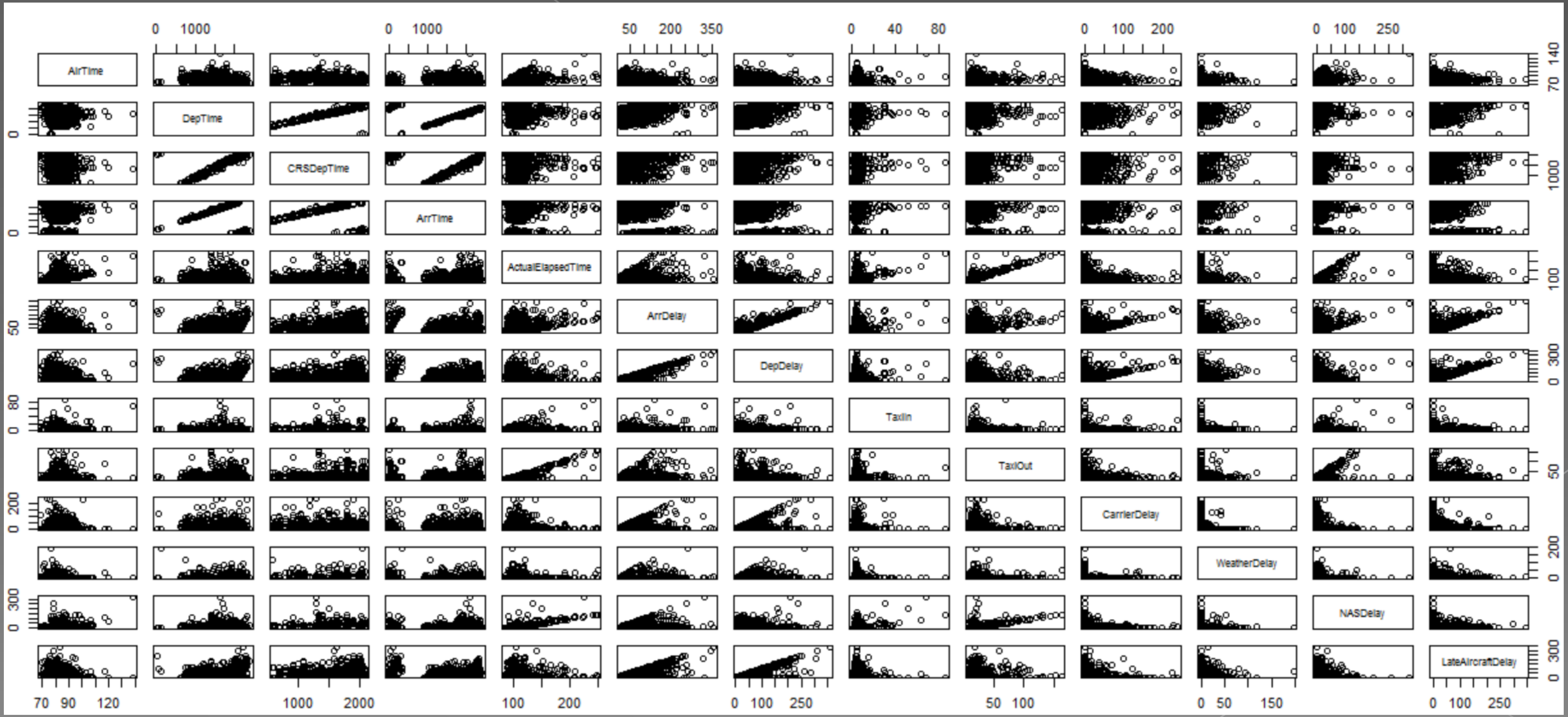
❖ Whereas, for Logistic regression, the dependent variable is categorical (i.e. qualitative data)

❖ In this case, I included the following as the dependent variable.

- ❖ For Multiple Regression – AirTime
- ❖ For Logistic Regression – WeatherDelay (For Delay 0, else 1)

Steps to Perform for Multiple Regression

- **Draw a scatter plot and Observe whether you can fit a line to describe a pattern**



Steps to Perform for Multiple Regression continued…

- Build a multiple linear regression model , $y = f(x) = \beta0 + \beta1X + e$

- fit_model <- lm(model$AirTime ~ as.factor(model$Month) + as.factor(model$DayofMonth) + as.factor(model$DayOfWeek) + model$DepTime + model$CRSDepTime + model$ArrTime + model$CRSArrTime + as.factor(model$UniqueCarrier) + as.factor(model$FlightNum) + model$ActualElapsedTime + model$CRSElapsedTime + model$ArrDelay + model$DepDelay + model$TaxiIn + model$TaxiOut + model$CarrierDelay + model$WeatherDelay + model$NASDelay + model$LateAircraftDelay)

Steps to Perform for Multiple Regression continued…

- Parameter Estimates

```
> # Dropping model$CarrierDelay
> fit_model <- lm(formula = model$AirTime ~
+                      +        model$ArrDelay + model$DepDelay +
+                      +        model$TaxiIn + model$TaxiOut)
> summary(fit_model)

Call:
lm(formula = model$AirTime ~ +model$ArrDelay + model$DepDelay +
    +model$TaxiIn + model$TaxiOut)

Residuals:
    Min      1Q  Median      3Q     Max
-7.1156 -1.6715 -0.3932  2.0813  8.4047

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    102.99754    0.38413  268.13   <2e-16 ***
model$ArrDelay   0.89767    0.01442   62.26   <2e-16 ***
model$DepDelay  -0.90124    0.01424  -63.28   <2e-16 ***
model$TaxiIn    -0.82366    0.02253  -36.56   <2e-16 ***
model$TaxiOut   -0.87367    0.01522  -57.40   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.866 on 1015 degrees of freedom
Multiple R-squared:  0.8025,	Adjusted R-squared:  0.8018
F-statistic:  1031 on 4 and 1015 DF,  p-value: < 2.2e-16
```

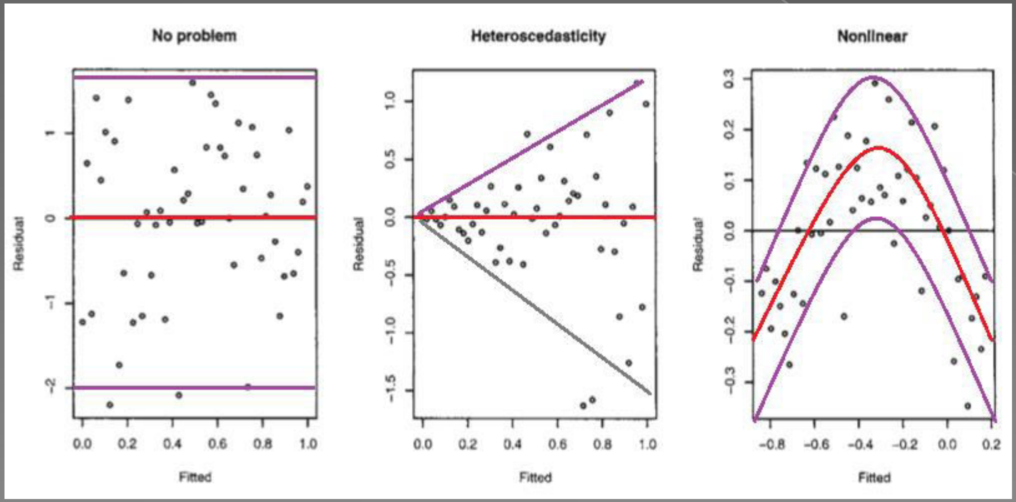
Steps to Perform for Multiple Regression continued…

- Goodness of Fit Test
  - A good model is determined by Goodness of fit.
  - There are 3 ways to determine the same.
  - These steps are to be performed sequentially.
    - 1. Overall Goodness using F-test
    - 2. Individual Parameter test
    - 3. Coefficient of determination (R2)

Steps to Perform for Multiple Regression continued…

- Residual Analysis
  - A residual plot is a scatter plot of the residuals against the predicted value.

Residual = Actual value – Predicted value



Goals in Residual Analysis

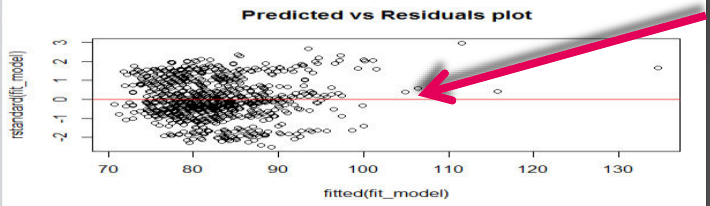
❖ Validate the constant variance

❖ Validate the linearity relationship

❖ Validate normal distribution of residuals (QQ-Plot)

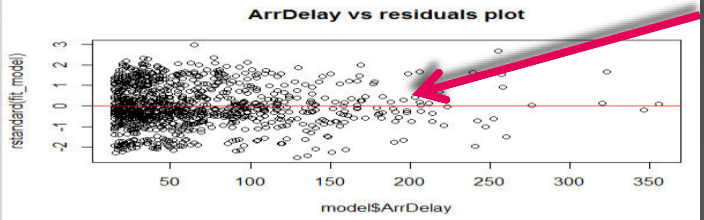
❖ Identify potential outliers (Cook's Distance)

```
#Residual Analysis
#Plot residuals vspredicted values
plot( fitted(fit_model), rstandard(fit_model), main="Predicted vs Residuals plot")
abline(a=0, b=0, col='red') #add zero line
```



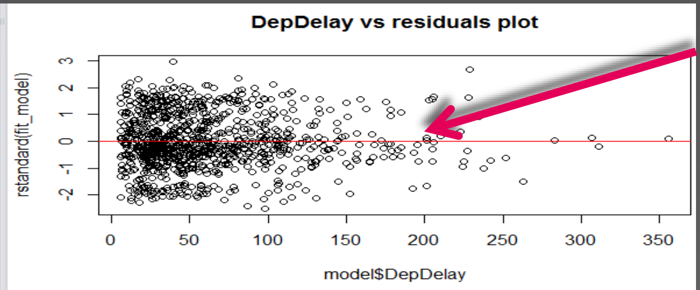
Plot Residuals vs Predicted values

```
> #Plot residuals vs each x-variables
> fit_model

Call:
lm(formula = model$AirTime ~ +model$ArrDelay + model$DepDelay +
    +model$TaxiIn + model$TaxiOut)

Coefficients:
   (Intercept)  model$ArrDelay  model$DepDelay    model$TaxiIn   model$TaxiOut
      102.9975          0.8977         -0.9012         -0.8237         -0.8737

> #Plot residuals vs ArrDelay
> plot(model$ArrDelay, rstandard(fit_model), main="ArrDelay vs residuals plot")
> abline(a=0, b=0, col='red') #add zero line
```



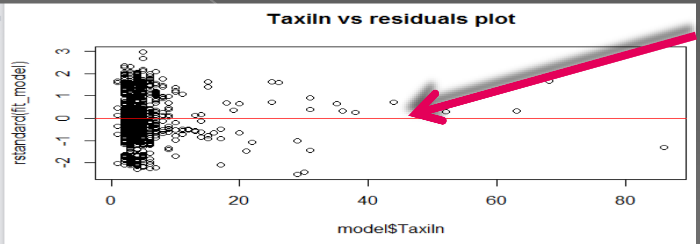
Plot Residuals vs ArrDelay

```
> #Plot residuals vs DepDelay
> plot(model$DepDelay, rstandard(fit_model), main="DepDelay vs residuals plot")
> abline(a=0, b=0, col='red') #add zero line
```



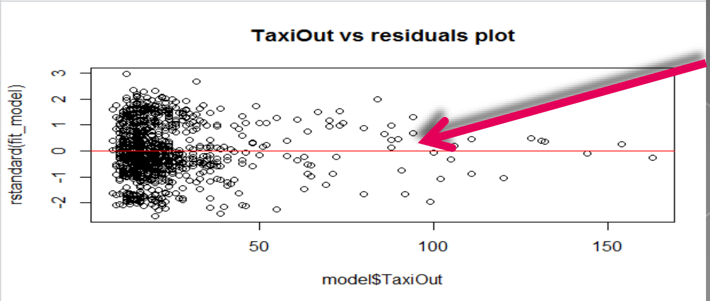
Plot Residuals vs DepDelay

```
> #Plot residuals vs TaxiIn
> plot(model$TaxiIn, rstandard(fit_model), main="TaxiIn vs residuals plot")
> abline(a=0, b=0, col='red') #add zero line
```



Plot Residuals vs TaxiIn

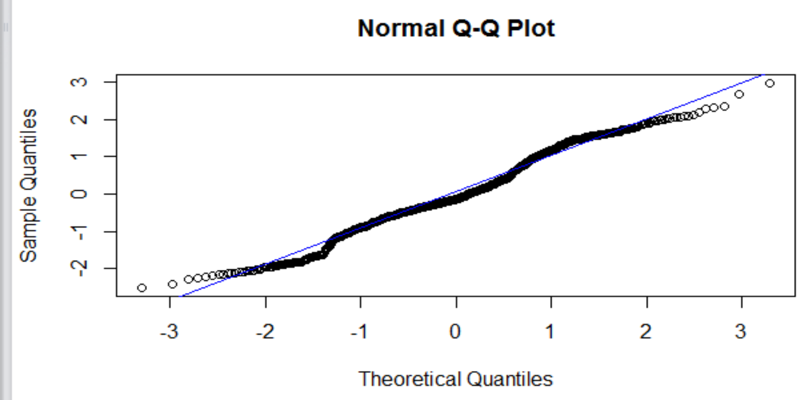
```
> #Plot residuals vs TaxiOut
> plot(model$TaxiOut, rstandard(fit_model), main="TaxiOut vs residuals plot")
> abline(a=0, b=0, col='red') #add zero line
```



Plot Residuals vs TaxiOut

QQ Plot and Outliers

```
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
>
> # QQ-Plot
> qq_plot <- (rstandard(fit_model))
> qqnorm(qq_plot)
> qqline(rstandard(fit_model), col='blue')
> 
```




```
>
> #Influential Points
> #Cook's Distance
> # n=1020, 4/n = 4/1020 = 0.003
> cook <- influence.measures(fit_model)
> summary(cook)
Potentially influential observations of
	 lm(formula = model$AirTime ~ model$ArrDelay + model$DepDelay +      model$TaxiIn + model$TaxiOut) :
```

	dfb.1_	dfb.m$AD	dfb.m$DD	dfb.m$TI	dfb.m$TO	dffit	cov.r	cook.d	hat
53	0.01	0.00	0.00	0.00	0.00	-0.03	1.04_*	0.00	0.03_*
63	0.04	0.01	-0.01	0.01	-0.06	-0.15	1.03_*	0.00	0.03_*
66	0.00	0.00	0.00	0.00	0.00	0.02	1.02_*	0.00	0.01
72	0.00	0.00	0.00	0.00	0.00	0.01	1.01_*	0.00	0.01
103	0.05	-0.03	0.02	-0.22	0.01	-0.36_*	1.00	0.03	0.02_*
115	0.02	-0.02	0.00	0.04	0.03	-0.21	1.01	0.01	0.02_*
121	0.01	-0.03	0.03	0.04	-0.04	-0.20	1.04_*	0.01	0.03_*

Regression Diagnostics

- R^2 / Adj - R^2 Coefficient of Determination
  - This is a measure of goodness of fit for a linear regression model.
  - Coefficient of determination is the calculation of the variation in the dependent variable to the variation in the independent variable. 0 means no linear relationship and 1 means perfect model. Usually R^2 value > 80% is considered a good model.
  - From the output, it is evident that 80.18% variation in Y is explained by X.

```
Residual standard error: 2.866 on 1015 degrees of freedom
Multiple R-squared:  0.8025,    Adjusted R-squared:  0.8018
```

  - The only difference between R^2 and Adjusted R^2 is that adjusted R^2 increases only when the new term added improves the model. Hence, it is more reliable than R^2

Taboo on R^2

- High R^2 means a better model ?
- Low R^2 means a bad model ?
- I found the above taboos a myth, since post execution of my model, I'm able to find Adj- R^2 =1 at a point. I had to recheck my X variables.
- Since they were highly collinear, I got 1 as the Adj- R^2 value, which is indeed not correct.

Multicollinearity

- ❖ Multicollinearity is an undesirable situation where the correlations among the independent variables are strong.
- ❖ When two X variables are highly collinear, be it negative or positive, they essentially convey the same information and when this happens, I found the regression results to be paradoxical.
- ❖ Situations that indicate Multicollinearity:
  - High F-test, but none of the X variables are significant.
  - Appearance of NA in the parameter estimates.

Problems due to Multicollinearity

- Multicollinearity misleadingly inflates the standard errors of coefficients.
- Thus, it makes some variables statistically insignificant while they should be otherwise significant.
- It is like two or more people are singing loudly at the same time. One cannot discern which is which. They offset each other.

How to detect Multicollinearity?

- Variation Inflation Factors (VIF) >= 5 indicates Multicollinearity.
- If there are two or more variables that has VIF greater than or around 5, one of the variables must be removed first.
- To determine the best one to remove, remove each one individually.
- Select the regression equation with highest R^2 .

Steps to Perform for Multiple Regression continued…

- Evaluations and Predictions
  - Hold Out Evaluation (Large Data)
  - N-Fold Evaluation
    (Small Data)
- I performed Hold Out Evaluation.

Model Selection

- Search Algorithms
  - Best Subset Regression
  - Backward Elimination
  - Stepwise Regression/Forward Selection
- Model Selection Methods
  - Adj-R^2
  - Mallow's Cp Statistics
  - AIC and BIC criterion
  - PRESS Statistics

```
> #Selecting the best model based on Adj-R2
> summary(fit_model)

Call:
lm(formula = model$AirTime ~ ++model$ArrDelay + model$DepDelay +
    ++model$TaxiIn + model$TaxiOut)

Residuals:
    Min      1Q  Median      3Q     Max
-7.1156 -1.6715 -0.3932  2.0813  8.4047

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    102.99754    0.38413  268.13   <2e-16 ***
model$ArrDelay   0.89767    0.01442   62.26   <2e-16 ***
model$DepDelay  -0.90124    0.01424  -63.28   <2e-16 ***
model$TaxiIn    -0.82366    0.02253  -36.56   <2e-16 ***
model$TaxiOut   -0.87367    0.01522  -57.40   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.866 on 1015 degrees of freedom
Multiple R-squared:  0.8025,   Adjusted R-squared:  0.8018
F-statistic:  1031 on 4 and 1015 DF,  p-value: < 2.2e-16

> #As Backward elimination has high Adj-R2(80.18%), so we take this model into consideration

> summary(fit_model2) #final reduced model by forward selection

Call:
lm(formula = model$AirTime ~ model$ActualElapsedTime + model$DepDelay +
    model$TaxiOut + model$ArrDelay)

Residuals:
    Min      1Q  Median      3Q     Max
-42.783  -1.536  -0.052   1.799  17.442

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)             21.66587    4.35185   4.979 7.52e-07 ***
model$ActualElapsedTime  0.67723    0.04124  16.422  < 2e-16 ***
model$DepDelay           0.12173    0.04268   2.852  0.00443 **
model$TaxiOut           -0.54940    0.01590 -34.550  < 2e-16 ***
model$ArrDelay          -0.13474    0.04261  -3.162  0.00161 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.878 on 1015 degrees of freedom
Multiple R-squared:  0.6386,    Adjusted R-squared:  0.6372
F-statistic: 448.3 on 4 and 1015 DF,  p-value: < 2.2e-16
```

```
> #Backward - Stepwise
> step(fit_model2, direction="backward", trace=TRUE)
Start:  AIC=2769.58
model$AirTime ~ model$ActualElapsedTime + model$DepDelay + model$TaxiOut +
    model$ArrDelay

                          Df Sum of Sq   RSS    AIC
<none>                                 15261 2769.6
- model$DepDelay           1     122.3 15383 2775.7
- model$ArrDelay           1     150.3 15411 2777.6
- model$ActualElapsedTime  1    4054.5 19315 3007.9
- model$TaxiOut            1   17947.0 33208 3560.6

Call:
lm(formula = model$AirTime ~ model$ActualElapsedTime + model$DepDelay +
    model$TaxiOut + model$ArrDelay)

Coefficients:
            (Intercept)  model$ActualElapsedTime           model$DepDelay            model$TaxiOut           model$ArrDelay
                21.6659                   0.6772                   0.1217                  -0.5494                  -0.1347
```

```
> > step(fit_model1,scope=list(upper=fit_model2, lower=~1), direction= "forward", trace=TRUE)
Start:  AIC=3619.02
model$AirTime ~ model$ActualElapsedTime

                 Df Sum of Sq   RSS    AIC
+ model$TaxiOut   1   19493.0 15809 2801.6
+ model$ArrDelay  1     982.0 34320 3592.2
+ model$DepDelay  1     857.7 34444 3595.9
<none>                        35302 3619.0

Step:  AIC=2801.6
model$AirTime ~ model$ActualElapsedTime + model$TaxiOut

                 Df Sum of Sq   RSS    AIC
+ model$ArrDelay  1    426.14 15383 2775.7
+ model$DepDelay  1    398.13 15411 2777.6
<none>                        15809 2801.6

Step:  AIC=2775.73
model$AirTime ~ model$ActualElapsedTime + model$TaxiOut + model$ArrDelay

                 Df Sum of Sq   RSS    AIC
+ model$DepDelay  1    122.31 15261 2769.6
<none>                        15383 2775.7

Step:  AIC=2769.58
model$AirTime ~ model$ActualElapsedTime + model$TaxiOut + model$ArrDelay +
    model$DepDelay


Call:
lm(formula = model$AirTime ~ model$ActualElapsedTime + model$TaxiOut +
    model$ArrDelay + model$DepDelay)

Coefficients:
```

Model Evaluation

- Hold Out Evaluation

```
> y1=predict.glm(fit_model,test.data)
Warning message:
'newdata' had 204 rows but variables found have 1020 rows
>
> y2=predict.glm(fit_model2,test.data)
Warning message:
'newdata' had 204 rows but variables found have 1020 rows
> y=test.data[,8]
>  rmse1 <- sqrt((y-y1)%*%(y-y1))/nrow(test.data)
> rmse2 <- sqrt((y-y2)%*%(y-y2))/nrow(test.data)
> rmse1
        [,1]
[1,] 1.34746
> rmse2
        [,1]
[1,] 1.28515
> 
```

Measuring Predictive Performance

Root Mean Square Error :

Best model minimizes RMSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{m}}$$

Mean Absolute Error

Best model minimizes MAE

$$MAE = \frac{\sum_{i=1}^{m}| y_i - \hat{y}_i |}{m}$$

Logistic Regression

- Predictor – Categorical or Numeric
- Response – Categorical
- Relationship between response(binary) and predictor(s)

Model for probability p=Pr(Y=1):

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + e$$

Steps to perform Logistic Regression

- Based on AIC or BIC value
- In this case, AIC value is used

```
> #Try with backward selection model
> step(full,direction = "backward",trace = F)

Call:  glm(formula = train_data$Wdelay ~ train_data$Month + train_data$DayofMonth +
    train_data$UniqueCarrier + train_data$FlightNum + train_data$ActualElapsedTime +
    train_data$CRSElapsedTime + train_data$ArrDelay + train_data$CarrierDelay +
    train_data$NASDelay, family = "binomial", data = train_data,
    control = list(maxit = 50))

Coefficients:
                (Intercept)             train_data$Month        train_data$DayofMonth
                  3.5873941                   -0.4126614                   -0.0202636
  train_data$UniqueCarrierMQ    train_data$UniqueCarrierUA         train_data$FlightNum
                 -2.5555662                   -0.6128026                    0.0005114
train_data$ActualElapsedTime     train_data$CRSElapsedTime          train_data$ArrDelay
                  0.0797740                   -0.1159837                    0.0100691
     train_data$CarrierDelay           train_data$NASDelay
                 -0.0361235                   -0.0844027

Degrees of Freedom: 815 Total (i.e. Null);  805 Residual
Null Deviance:      637.8
Residual Deviance: 498.8        AIC: 520.8
```

Steps to perform Logistic Regression contd.

```
> full1 <- glm(train_data$Wdelay ~ train_data$Month+train_data$ArrDelay+train_data$DepDelay+train_data$Carrier
Delay+train_data$NASDelay,data = train_data, family = "binomial", control = list(maxit = 50))
> summary(full1)

Call:
glm(formula = train_data$Wdelay ~ train_data$Month + train_data$ArrDelay +
    train_data$DepDelay + train_data$CarrierDelay + train_data$NASDelay,
    family = "binomial", data = train_data, control = list(maxit = 50))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0153  -0.5361  -0.3468  -0.1928   3.2395

Coefficients:
                         Estimate Std. Error z value Pr(>|z|)
(Intercept)             -0.430183   0.291422  -1.476 0.139904
train_data$Month        -0.433393   0.071232  -6.084 1.17e-09 ***
train_data$ArrDelay      0.081100   0.021615   3.752 0.000175 ***
train_data$DepDelay     -0.071429   0.021580  -3.310 0.000933 ***
train_data$CarrierDelay -0.036328   0.008238  -4.410 1.03e-05 ***
train_data$NASDelay     -0.073820   0.022929  -3.219 0.001284 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 637.84  on 815  degrees of freedom
Residual deviance: 521.19  on 810  degrees of freedom
AIC: 533.19

Number of Fisher Scoring iterations: 6
```

Steps to perform Logistic Regression contd.

- McFadden's R-square

```
> mcFmodel <- glm(formula = train_data$Wdelay ~ train_data$Month + train_data$DayofMonth + train_data$UniqueCarrier + train_data$FlightNum + train_data$ActualElapsedTime + train_data$CRSElapsedTime + train_data$ArrDelay + train_data$CarrierDelay + train_data$NASDelay, family = "binomial", data = train_data, control = list(maxit = 50))
> mcFnmodel <- glm(train_data$Wdelay ~ 1, family = "binomial")
> 1-logLik(mcFmodel)/logLik(mcFnmodel)
'log Lik.' 0.2179834 (df=11)
> |
```

- 95% CI for the coefficients

```
> #Confidence Intervals For Model Coefficients
> confint(mcFmodel)
Waiting for profiling to be done...
                                    2.5 %       97.5 %
(Intercept)                  -7.0824671151 14.272594321
train_data$Month             -0.5691316353 -0.263144833
train_data$DayofMonth        -0.0452549713  0.004323568
train_data$UniqueCarrierMQ   -5.0361771094 -0.315940003
train_data$UniqueCarrierUA   -1.2386408595  0.023931861
train_data$FlightNum         -0.0001104443  0.001136609
train_data$ActualElapsedTime  0.0366785209  0.126898499
train_data$CRSElapsedTime    -0.2250645625 -0.008808043
train_data$ArrDelay           0.0057002817  0.014555780
train_data$CarrierDelay      -0.0539527266 -0.021690803
train_data$NASDelay          -0.1349492785 -0.038537181
```

Steps to perform Logistic Regression contd.

- Computing exp(coefficients) to analyze change in odds for changes in X

```
> exp(coef(mcFmodel))
                (Intercept)              train_data$Month        train_data$DayofMonth
                36.13977707                    0.66188636                   0.97994035
  train_data$UniqueCarrierMQ    train_data$UniqueCarrierUA          train_data$FlightNum
                 0.07764825                    0.54183018                   1.00051151
train_data$ActualElapsedTime     train_data$CRSElapsedTime           train_data$ArrDelay
                 1.08304225                    0.89048976                   1.01011998
     train_data$CarrierDelay            train_data$NASDelay
                 0.96452116                    0.91906105
>
```

Steps to perform Logistic Regression contd.

- Change in Odds

```
> exp(confint(mcFmodel))
Waiting for profiling to be done...
                                   2.5 %       97.5 %
(Intercept)                  0.000839699 1.579461e+06
train_data$Month             0.566016734 7.686306e-01
train_data$DayofMonth        0.955753761 1.004333e+00
train_data$UniqueCarrierMQ   0.006498544 7.291032e-01
train_data$UniqueCarrierUA   0.289777799 1.024221e+00
train_data$FlightNum         0.999889562 1.001137e+00
train_data$ActualElapsedTime 1.037359478 1.135302e+00
train_data$CRSElapsedTime    0.798464666 9.912306e-01
train_data$ArrDelay          1.005716559 1.014662e+00
train_data$CarrierDelay      0.947476896 9.785428e-01
train_data$NASDelay          0.873760229 9.621959e-01
```

Steps to perform Logistic Regression contd.

- VIF

```
> vif(mcFmodel)
           GVIF Df GVIF^(1/(2*Df))
mon    1.179723  1        1.086150
dom    1.020857  1        1.010375
uq     3.713334  2        1.388165
fn     3.336038  1        1.826482
aet   17.387084  1        4.169782
crset  1.330932  1        1.153660
ad     1.172485  1        1.082813
cd     1.057806  1        1.028497
nasd  16.854155  1        4.105381
```

Steps to perform Logistic Regression contd.

- Sensing how strong the predictor is

```
>
> set.seed(1021)
> n <- 1020
> x <- 1*(runif(n)<0.5)
> pr <- (x==1)*0.9+(x==0)*0.1
> y <- 1*(runif(n) < pr)
> mod <- glm(y~x, family="binomial")
> nullmod <- glm(y~1, family="binomial")
> 1-logLik(mod)/logLik(nullmod)
'log Lik.' 0.4928337 (df=2)
> |
```

```
> set.seed(1021)
> n <- 1020
> x <- 1*(runif(n)<0.5)
> pr <- (x==1)*0.99+(x==0)*0.01
> y <- 1*(runif(n) < pr)
> mod <- glm(y~x, family="binomial")
> nullmod <- glm(y~1, family="binomial")
> 1-logLik(mod)/logLik(nullmod)
'log Lik.' 0.8893839 (df=2)
```

```
> set.seed(1021)
> n <- 1020
> x <- 1*(runif(n)<0.5)
> x <- 1*(runif(n)>0.5)
> pr <- (x==1)*0.9+(x==0)*0.1
> y <- 1*(runif(n) < pr)
> mod <- glm(y~x, family="binomial")
> nullmod <- glm(y~1, family="binomial")
> 1-logLik(mod)/logLik(nullmod)
'log Lik.' 0.4765672 (df=2)
> |
```

Steps to perform Logistic Regression contd.

- Predicting values based on equation

```
> pred_dataglm <- data.frame(mon=6,dom=30,uq="AA",fn=1448,aet=101,crset=105,ad=39,cd=10,nasd=0)
> predict(mcFmodel, pred_dataglm,se.fit = TRUE,interval=c("none","confidence","prediction"), level=0.95,type="
response")
$fit
         1
0.05490634

$se.fit
         1
0.02068382

$residual.scale
[1] 1

> predict(mcFmodel, pred_dataglm, type="response")
         1
0.05490634
> pred_dataglm <- data.frame(mon=6,dom=5,uq="AA",fn=580,aet=98,crset=105,ad=120,cd=0,nasd=0)
> predict(mcFmodel, pred_dataglm, type="response")
        1
0.1364101
> pred_dataglm <- data.frame(mon=1,dom=3,uq="UA",fn=602,aet=102,crset=103,ad=24,cd=0,nasd=0)
> predict(mcFmodel, pred_dataglm, type="response")
        1
0.3189236
> pred_dataglm <- data.frame(mon=5,dom=2,uq="MQ",fn=3908,aet=129,crset=100,ad=160,cd=0,nasd=29)
> predict(mcFmodel, pred_dataglm, type="response")
        1
0.2283462
> pred_dataglm <- data.frame(mon=3,dom=28,uq="UA",fn=622,aet=95,crset=106,ad=42,cd=0,nasd=0)
> predict(mcFmodel, pred_dataglm, type="response")
         1
0.05702596
>
>
```