# **STATS FINAL PROJECT APPENDIX**

# **Project Title: Data Analysis of Different Cars from Dekho Website using Regression Analysis and time series forecast.**

# 

# Submitted By

1)Sai Lokesh Siddanathi

2) Kore Kiran

A yellow and black logo

Description automatically generated with low confidence

DATE : 10TH DEC,2022

**University of Colorado Denver**

**(Aug 2022-May 2024)**

**Introduction:**

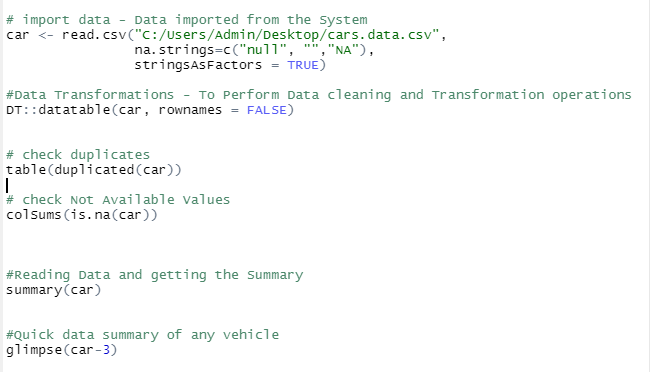
* The goal of this research is to create algorithms that anticipate automotive selling prices.
* We examined the outcomes of three different models for predicting automobile selling prices. Every automobile brand manufactures various types of cars, with a brand describing the kind of car (think of Honda Civic vs Honda Odyssey, one a compact car and the other a minivan). This dataset contains around 30 models.

**DATA SET:**

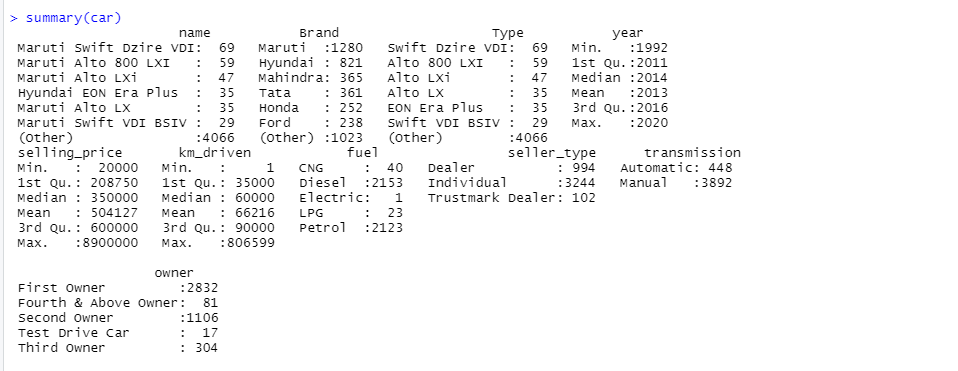
* Name: Name of the car
* Brand: Producer of the car
* Type: Version of the car
* Selling price: Selling Price of the car
* Km driven: Kilometres driven by the car
* Fuel type: Type of the Fuel
* Seller type: Seller who is individual or Dealer
* Transmission: Automatic/Manual
* Owner: Number of previous owners of the car

**Data Import and Connection Establish:**

Below are the steps performed to import the data to perform the data analysis in R.

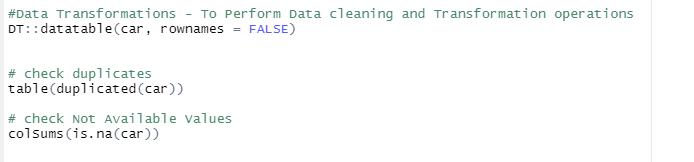


**Summary of the Data:**



**Data Exploration, Data Cleaning and Data Transformation:**

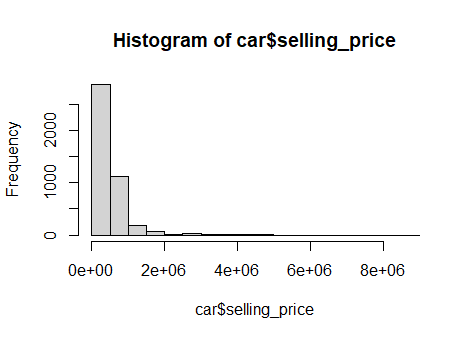
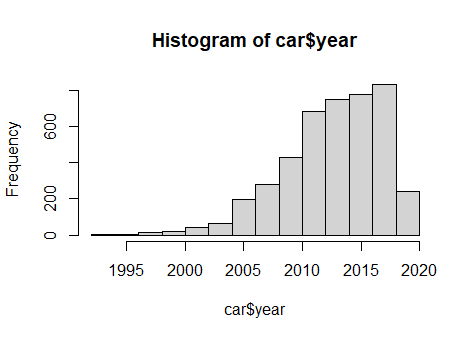
Here the Data is cleaned and transformed for making it usable for the Analysis. Here the steps are performed to clean the file, check duplicates and the clean the missing data.

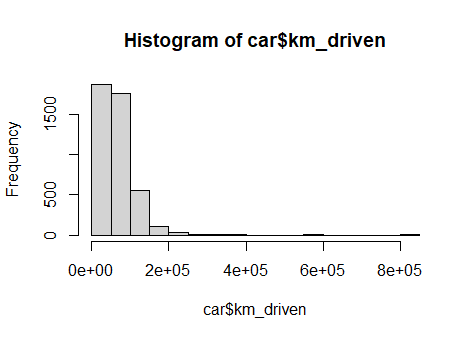


Steps are taken to make sure that there are no missing values anymore after this step.

**Dependency Check for Normal Distribution:**

Dependency Check has been done on the important parameters for the normal Distribution. Images are printed below for the reference.





**Linearity Check and Correlation:**

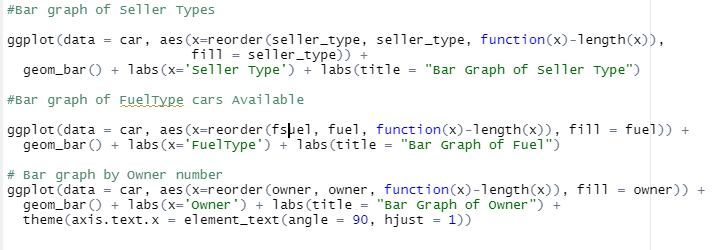
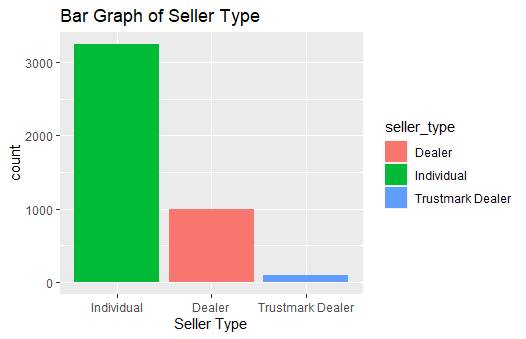
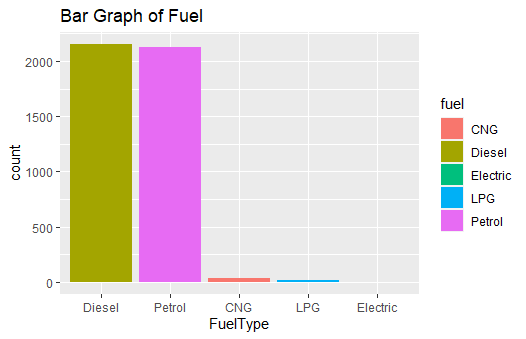
Here we Checking the linearity between the dependent and Linear variable. Linearity can be checked from the residual and Fitted Plot.

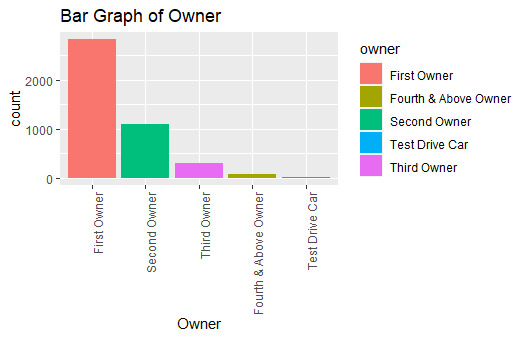
**Correlation:** Correlation analysis measures how two variables are related. Thecorrelation coefficient (r) is a statistic that tells you the strengthand direction of that relationship. It is expressed as a positive ornegative number between -1 and 1.



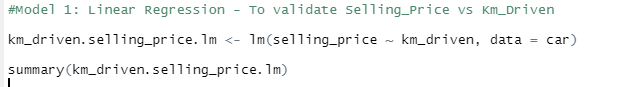
## Plotting categorical Values and checking for distribution:

## Code is written to depict the pictorial representation of the existing data for all the parameters.

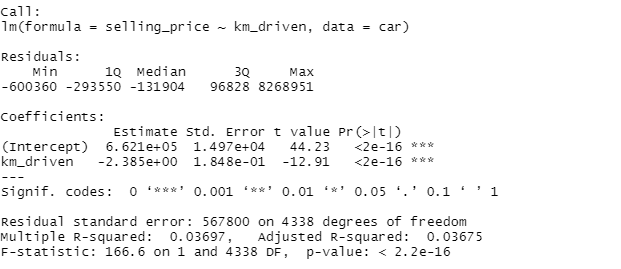




## Linear Regression:



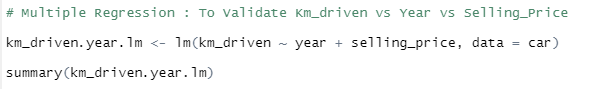
### Result of Linear Regression:



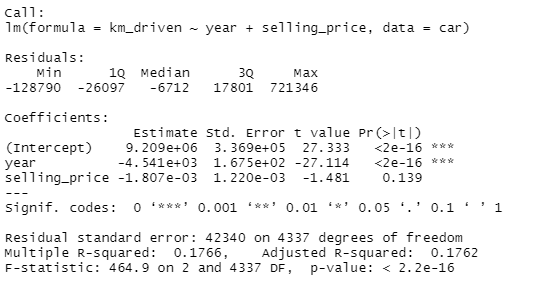
**Conclusion:**

The considerable thing to be observed here is the p value is <2.2e-16) which says whether the model fits the data well or not and the Value of y-intercept is 6.621. The Estimated effect on Kms driven is -2.385. The T-value is -12.91

## Multiple Regression:



### Result of Multiple Regression:

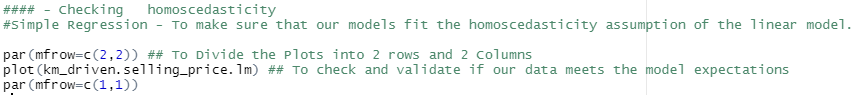


**Conclusion:**

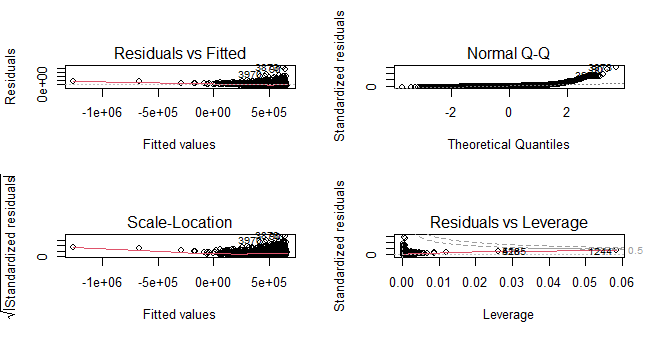
* The estimated effect of Year on kilometers Driven is -0.00454,
* while The estimated effect of Selling Price on -0.001807
* The standard errors for these both are very high and t Values are very small(-27 and -1 respectively)
* P Value refers these small errors and the high T statistics. Remember that somehow this data is composed for this example. so these correlations would not be nearly as evident in real life!

## Checking homoscedasticity:

**Simple Regression:** To make sure that our models fit the homoscedasticity assumption of the linear model.

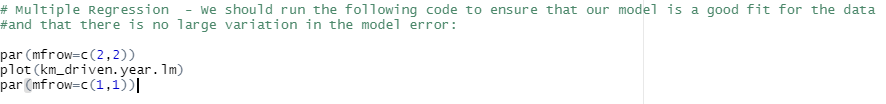


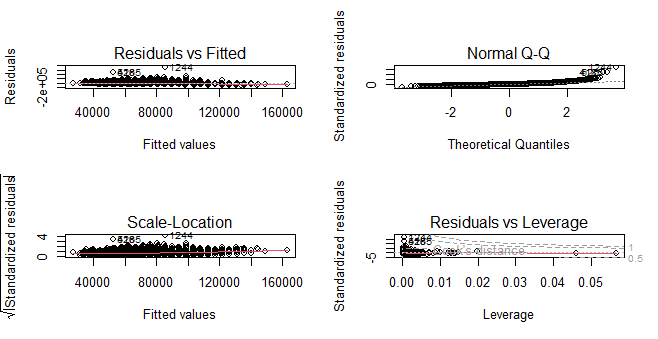
### Simple Regression- Result:



### Multiple Regression - Result:

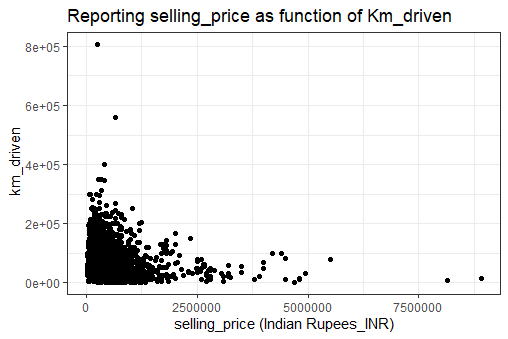
We should run the following code to ensure that our model is a good fit for the data and that there is no large variation in the model error

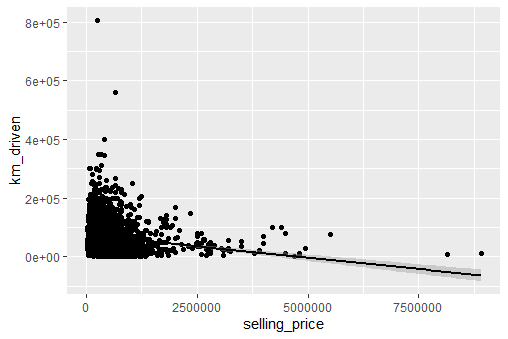




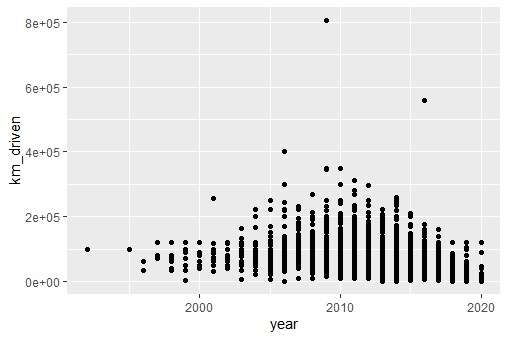
## Final Plot - Visuals:

**Simple Regression:** With the help of Simple regression, we found the values and plotted the graph to see the values. Below is the graph that turned as the output.





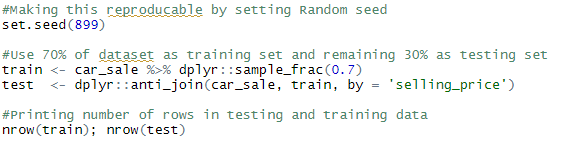
**Note: Added a Linear Regression Line to this model after computing the normal data.**



To Conclude, We are able to visualize from this models that the price is varied with the Km\_Driven and the Km-Driven also has a relation with the years.

## Time Series Forecast:

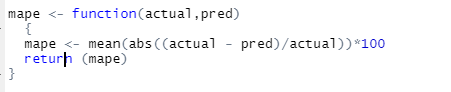
Here we have cleaned the data and setup the random seed. Later we have splitted the data into Testing and Training data.



### Mean Absolute Percentage Error (MAPE):

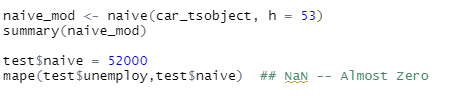
This is to create a utility function for calculating Mean Absolute Percentage Error (or MAPE), which shall be used to evaluate the performance of the forecasting models.

Here Is the code that is written for calculating MAPE.

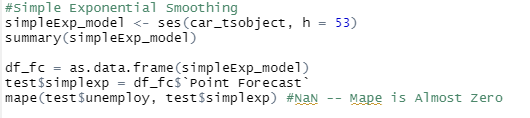


We shall be using the MAPE from the above to calculate all the parameters.

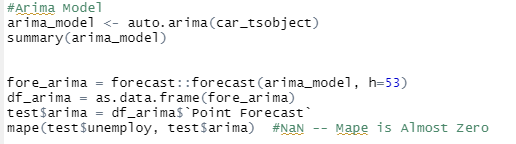
### Naive Forecasting Method:



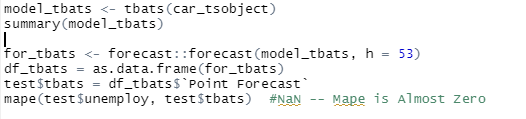
### Simple Exponential Smoothing:



### Arima Model:



### TBATS – Model:



Generally, Lower the MAPE better is the performance of the model.

R CODE:

import libraries

library(tidyverse)

library(dplyr)

library(inspectdf)

library(magicfor)

library(GGally)

library(ggplot2)

library(viridis)

library(MLmetrics)

library(car)

library(caret)

library(vtreat)

library(lmtest)

library(pastecs)

library(magrittr)

library(ggplot2)

library(dplyr)

library(broom)

library(ggpubr)

library(forecast)

library(tidyverse)

library(fpp)

library(forecast)

library(backtest)

library(quantmod)

library(lubridate)

library(sos)

library(readr)

library(ggplot2)

library(forecast)

library(fpp2)

library(TTR)

library(dplyr)

findFn("starts\_with")

# import data - Data imported from the System

car <- read.csv("/Users/sailokeshsiddanathi/Documents/stats project/CarFinal.csv", na.strings=c("null", "","NA"), stringsAsFactors = TRUE)

#Data Transformations

DT::datatable(car, rownames = FALSE)

# check duplicates

table(duplicated(car))

# check Not Available Values

colSums(is.na(car))

#Reading Data and getting the Summary

summary(car)

#Checking if our dependent variable follows a Normal Distribution

hist(car$selling\_price)

hist(car$year)

hist(car$km\_driven)

#Checking the linearity between the dependent and Linear variable

plot(selling\_price ~ year, data = car)

#Checking the correlation

cor(car$year,car$selling\_price)

#Bar graph of Seller Types

ggplot(data = car, aes(x=reorder(seller\_type, seller\_type, function(x)-length(x)),

fill = seller\_type)) +

geom\_bar() + labs(x='Seller Type') + labs(title = "Bar Graph of Seller Type")

#Bar graph of FuelType cars Available

ggplot(data = car, aes(x=reorder(fuel, fuel, function(x)-length(x)), fill = fuel)) +

geom\_bar() + labs(x='FuelType') + labs(title = "Bar Graph of Fuel")

# Bar graph by Owner number

ggplot(data = car, aes(x=reorder(owner, owner, function(x)-length(x)), fill = owner)) +

geom\_bar() + labs(x='Owner') + labs(title = "Bar Graph of Owner") +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))

#Model 1: Linear Regression - To validate Selling\_Price vs Km\_Driven

km\_driven.selling\_price.lm <- lm(selling\_price ~ km\_driven, data = car)

summary(km\_driven.selling\_price.lm)

# Multiple Regression : To Validate Km\_driven vs Year vs Selling\_Price

km\_driven.year.lm <- lm(km\_driven ~ year + selling\_price, data = car)

summary(km\_driven.year.lm)

#### - Checking homoscedasticity

#Simple Regression - To make sure that our models fit the homoscedasticity assumption of the linear model.

par(mfrow=c(2,2))

## To Divide the Plots into 2 rows and 2 Columns

plot(km\_driven.selling\_price.lm) ## To check and validate if our data meets the model expectations

par(mfrow=c(1,1))

#Visualizations: Simple Regression - Plotting the data on graph

year.graph<-ggplot(car, aes(x=selling\_price, y=km\_driven))+

geom\_point()

year.graph

## Adding Linear regression Line to Plotted Data

year.graph <- year.graph + geom\_smooth(method="lm", col="black")

# geom\_smooth -- for adding Regression Line

year.graph

## Adding equation for regression Line

year.graph <- year.graph + stat\_regline\_equation(label.x = 5, label.y = 7)

year.graph

#

year.graph +

theme\_bw() +

labs(title = "Reporting selling\_price as function of Km\_driven",

x = "selling\_price (Indian Rupees\_INR)",

y = " km\_driven")

## Visualization: For Multiple Regression

plottingnew.data <- expand.grid(km\_driven = seq(min(car$km\_driven), max(car$km\_driven), length.out=30),

Year=c(min(car$year), mean(car$year), max(car$year)))

plottingnew.data$km\_driven <- round(plottingnew.data$km\_driven, digits = 5)

plottingnew.data$km\_driven <- as.factor(plottingnew.data$km\_driven)

# Plotting the final data

km\_driven.plot <- ggplot(car, aes(x=year, y=km\_driven)) +

geom\_point()

km\_driven.plot

#Time Series Forecast

car\_sale <- read.csv("C:/Users/Admin/Desktop/cars.data.csv",

na.strings=c("null", "","NA"),

stringsAsFactors = TRUE)

glimpse(car\_sale)

#Creating ID Variable

car\_sale$id <- 1:nrow(car\_sale)

#Making this reproducable by setting Random seed

set.seed(899)

#Use 70% of dataset as training set and remaining 30% as testing set

train <- car\_sale %>% dplyr::sample\_frac(0.7)

test <- dplyr::anti\_join(car\_sale, train, by = 'selling\_price')

#Printing number of rows in testing and training data

nrow(train); nrow(test)

car\_tsobject <- ts(train[, 6], start = c(1950, 1), end = c(2022, 12), frequency = 12)

# create a utility function for calculating Mean Absolute Percentage Error (or MAPE),

mape <- function(actual,pred)

{

mape <- mean(abs((actual - pred)/actual))\*100

return (mape)

}

#Naive Forecasting Method

naive\_mod <- naive(car\_tsobject, h = 53)

summary(naive\_mod)

test$naive = 52000

mape(test$unemploy,test$naive) ## NaN -- Almost Zero

#Simple Exponential Smoothing

simpleExp\_model <- ses(car\_tsobject, h = 53)

summary(simpleExp\_model)

df\_fc = as.data.frame(simpleExp\_model)

test$simplexp = df\_fc$`Point Forecast`

mape(test$unemploy, test$simplexp) #NaN -- Mape is Almost Zero

#Arima Model

arima\_model <- auto.arima(car\_tsobject)

summary(arima\_model)

fore\_arima = forecast::forecast(arima\_model, h=53)

df\_arima = as.data.frame(fore\_arima)

test$arima = df\_arima$`Point Forecast`

mape(test$unemploy, test$arima) #NaN -- Mape is Almost Zero

#TBATS -- Model

model\_tbats <- tbats(car\_tsobject)

summary(model\_tbats)

for\_tbats <- forecast::forecast(model\_tbats, h = 53)

df\_tbats = as.data.frame(for\_tbats)

test$tbats = df\_tbats$`Point Forecast`

mape(test$unemploy, test$tbats) #NaN -- Mape is Almost Zero