Tutorial # 2

As we discussed in the last tutorial, we will use the scientific approach for forecasting i.e., statistics. And the available state-of-the-art technology is machine learning. In this data-driven approach, we first cover some fundamental concepts before we can use these fantastic tools that machine learning has to offer.

Machine Learning Algorithms

Imagine Ahmad, who wants to learn driving a car. He hires you to teach him driving. How would you teach him? He will need to understand the rules of traffic and what he should do in various situations that he might encounter while driving on his own. Machine learning algorithms work similarly. We need to teach the algorithm various situations in which we want it to make reasonable decisions.

Collecting Data

To teach Ahmad about various situations that he might encounter while driving in real life, you will have to create as many traffic situations as you can think of so that no situation is new for him or he will not know what to do in that situation.

In the same manner, we need to collect diverse data to teach any computing machine how something in real life works. One such example is PRECON. EIG collected this data by surveying various households and selecting a diverse group of households to install smart meters. This energy consumption data collected from each household will help machine learning algorithms understand how the electricity consumption patterns vary on a regular day or some special day e.g., rainy day, public holiday, or some religious event. This, in turn, will help computers forecast energy demand, which is our ultimate goal.

Preprocessing

Maybe some of the situations that you thought of, to teach Ahmad driving are just ridiculous, and you think they would never occur in real life. Those situations should be skipped. Some situations might require modification.

Data that is collected for machine learning also requires some cleaning and pre-processing before we can use it for training our model.

Processing

Its time to train Ahmad and guide him through all the situations, in the hope that once he completes it all, he is ready for real-life driving. It depends on the situations you created and the intelligence of Ahmad, whether he becomes a good driver at the end of the training or not.

The machine learning algorithm also goes through the same process of learning. Although the learning process for a computer is much faster, thanks to the high processing power available, but it all depends on the data you collected and the sophistication of your model that how good the model performs.

Testing

After the training is completed, both Ahmad and the machine learning algorithm should go through rigorous testing to check whether they are ready for real-life or not.

We do this by hiding some of the situations from the training and then testing our model on these unseen data.

Self Driving Cars

Several companies all over the world are working on building self-driving cars (see the figure below). One way to look at these autonomous vehicles is as a machine learning algorithm. In fact, every car has a pre-trained algorithm that knows various scenarios that might occur on the road and what it should do in each of those situations.

One of the decisions that a self-driving car has to make is to decide a safe distance that it should apply the brake to avoid colliding with the vehicle in front. The braking distance depends on the speed of the car, as a car at high speed should apply brake early, or it is at risk of causing an accident as it would travel a greater braking distance.

So the question is if the speed of the car is known, what is the shortest distance that the brake should be applied to avoid a collision?

To develop such a model, we will use a dataset already available in R.

The following code shows the first few rows of the data. The data frame contains two columns: speed and dist. You can learn more about the data by typing **?cars** in Console.

```
head(cars)
```

```
##        speed dist
## 1 4 2 ## 2 4 10 ##
3 7 4

## 4 7 22 ## 5 8 16
## 6 9 10
```

The following code shows a summary of the two columns. The maximum speed recorded is only 25 mph, which is understandable as the data was collected in the 1920s.

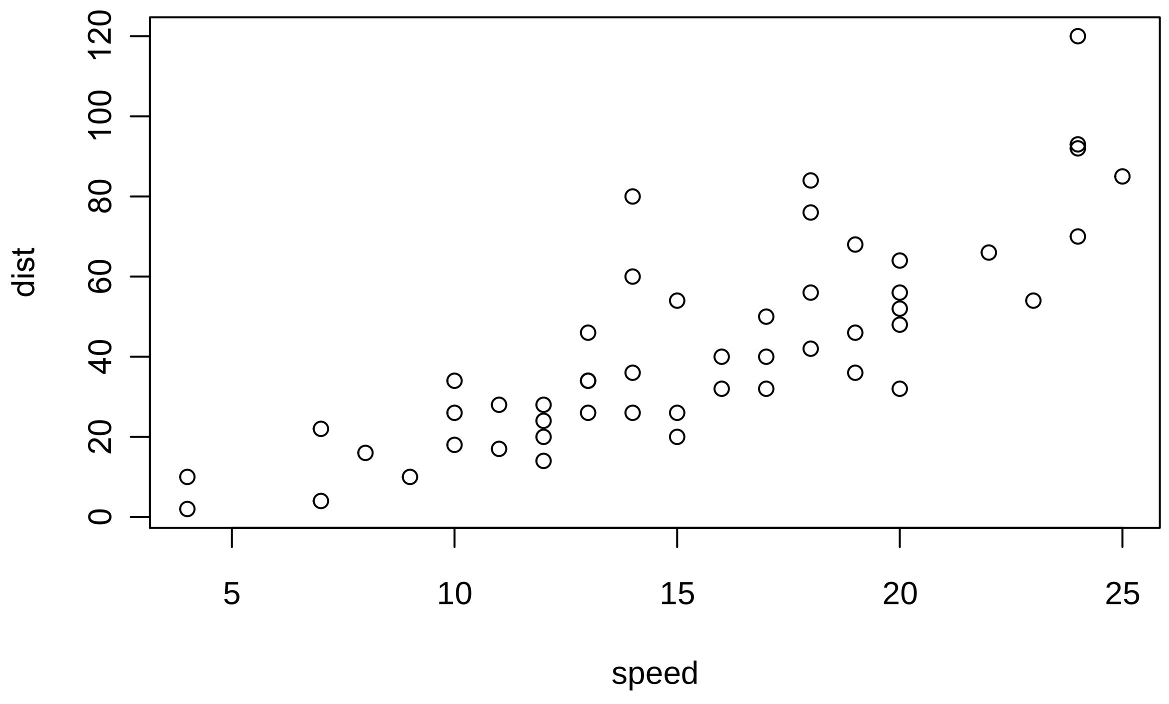
```
summary(cars)
```

```
##          speed               dist

## Min.      : 4.0     Min.     : 2.00
## 1st Qu.:12.0        1st Qu.: 26.00
## Median :15.0        Median : 36.00
## Mean   :15.4        Mean   : 42.98
## 3rd Qu.:19.0        3rd Qu.: 56.00
## Max.   :25.0        Max.   :120.00
```

The plot below shows dist vs. speed. As expected, there seems to be a positive correlation between the two. If a car applies brakes at higher speeds, it will travel a greater distance.

```
plot(cars)
```



The Pearson-correlation coefficient for the two columns is 0.806, which shows a strong linear positive relation. Due to this strong linear relation, I have decided to train a linear regression model.

```
cor(cars$speed, cars$dist)
```

```
## [1] 0.8068949
```

Before applying the linear regression, we should divide our data into train and test subsets. Usually, this means 80% of data is used to train the model, and the other 20% is used to test the model.

The code below stores the training data in *train_cars* and the testing data in *test_cars*. See if you can understand the rest of the code.

To get help on any function, click on the function and press F1

```
smp_size <- floor(0.80 * nrow(cars))

train_ind <- sample(seq_len(nrow(cars)), size = smp_size)


train_cars <- cars[train_ind, ] test_cars <- cars[-
train_ind, ]
```

Now we come to the training part. The *lm()* function is used to train a linear regression model. "~" is used to specify that we want the linear model to learn the relation in which *speed* is known, and *dist* needs to be predicted.

The linear regression model means fitting the below equation. What we need from our computing machines is to calculate β_0 and β_1, which are the two unknown coefficients of the linear equation.

$$dist = \beta_0 + \beta_1 * speed$$

The below line of code will calculate the two coefficients. Reading the below line of code in plain english: Predict *dist* using *speed* and find data for both variables in *train_cars* data frame and save the results in *lm_model*.

```
lm_model = lm(dist~speed, data = train_cars) lm_model
```

```
##
## Call:
## lm(formula = dist ~ speed, data = train_cars)
##
## Coefficients:
## (Intercept)                  speed
##           -20.612            4.145
```

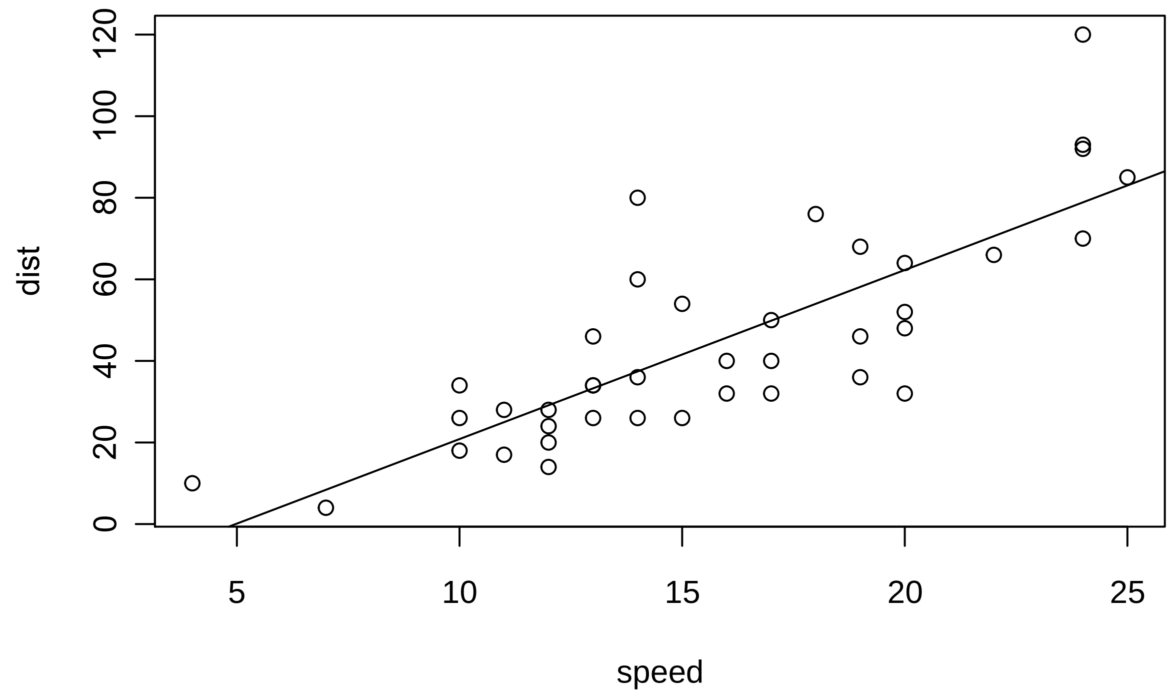
The results of the linear regression model show the value of the two coefficients. We can interpret the results as follows:

1. For every one mph increase in speed, the car will travel an extra 1 feet on average.
2. If the speed is zero, the car will travel 14.134 ft in the reverse direction. (This does not make sense) Given the data in our training data frame (*train_cars*), this is the best that the linear regression model could learn. It does not understand what would happen when the car is stationary.

$$dist = -14.134 + 3.838 * speed$$

The plot below shows the training data and the regression line calculated by the linear model.

```
plot(train_cars) abline(lm_model)
```



The summary of the linear model shows several interesting information about the linear model. Learn about this from this link.

Email me the interpretation of this summary.

```
summary(lm_model)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = train_cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.292  -9.965  -2.131  10.498  42.579
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -20.6116     8.2275  -2.505   0.0166 *
## speed         4.1452     0.4986   8.314 4.44e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.52 on 38 degrees of freedom
## Multiple R-squared:  0.6453, Adjusted R-squared:  0.6359
## F-statistic: 69.12 on 1 and 38 DF,  p-value: 4.435e-10
```

The only thing left to do is to test the model. The following code predicts the *dist* column from the testing data.

```
predict(lm_model, test_cars)
```

```
##         1           4           5           6           24          32          33     ##     -4.030841      8.404693
12.549871 16.695049 41.566117 54.001650 54.001650
##                 35                 42                 45
## 54.001650 62.292006 74.727540
```

To check how the model performed, statisticians have devised some performance metrics such as RMSE (Root mean square error). There is a package called “forecast* in R which can calculate all these metrics at once. You can install the package using”install.packages(“forecast”)".

```
# Load the package library(forecast)
```

```
## Registered S3 method overwritten by 'xts':
##      method          from
##         as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##      method                    from
##         as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##      method                       from
##          fitted.fracdiff              fracdiff
##             residuals.fracdiff fracdiff
```

```
# Calculating error metrics accuracy(test_cars$dist,

predict(lm_model, test_cars))
```

```
##                             ME       RMSE          MAE          MPE        MAPE
## Test set 1.220939 15.0339 12.23553 5.316504 55.01674
```

The *accuracy* function calculates five error metrics. ME (Mean Error) means that the model is off by 1.2209386 ft on average, which given the application of this model is not a good number. Self driving cars should be accurate enough to avoid any collision.

Email me the interpreation of all the error metrics and what it implies in case of our application of self driving cars.

You can find several great resources online explaining linear regression and how to build a model using it in R.

In later tutorials we will use linear regression to create a model for STLF too.

Cereals is a very popular dataset used to learn linear regression. Build a linear regression model to predict the rating of a cereal using only one predictor e.g. calories, fiber or carbo for this dataset. Follow the complete procedure of training and testing.

Email me your detailed report.