Titanic Data Analysis

```
library(tufte)
```

Data Analysis using Titanic Dataset

To summarize in bullet points, there are following merits.

- There is a lot of good literature for data visualization and analysis.
- With ggplot, we can create high quality plots without tuning.
- ggplot has a consistent API and is easy to use. The ggplot object and the geom object allow for a very good syntax and abstraction of the plots.
- With dplyr, it is easy to write complex data wranglings. The Pipe operator can be used to concisely describe the data wrangling process to be followed.

Using R, we can easily perform beautiful plots and analysis using a very well-developed API. We can also refer to the literature of good statistical users who are using R as their main tool.

```
library(caret) # Data Splitting
library(lightgbm) # lightbgm model
library(knitr) # report generation
library(ggrepel) # label
library(OneR) # binning
library(tidyverse) # metapackage of all tidyverse packages
set.seed(123)
```

```
search()
```

Load data and see overview

```
train_path <- "/Users/amanda/College Mini/titanic/train.csv"
test_path <- "/Users/amanda/College Mini/titanic/test.csv"
sub_path <- "/Users/amanda/College Mini/titanic/gender_submission.csv"

train <- read_csv(file = train_path)
test <- read_csv(file = test_path)
sub <- read_csv(file = sub_path)
```

read_csv returns tibble, not data frame. Tibbles are a modern take on data frames.

In particular, the section on Tibbles vs data frames contains examples of the differences between tibbles and data frames.
We can use tibble like pandas DataFrame. But there are some difference. For examle , when we take subset by [], tibble returns tibble. If we use pandas dataframe, it sometimes returns pandas Series, so we may use different API between dataframe and series. The R data frame may also return a vector as well.

```
# Tibble returns tibble.
train[,c("Pclass")]
```

Pclass <dbl>
3
1
3
1
3
3
1
3
3
2

1-10 of 891 rows Previous 1 2 3 4 5 6 … 90 Next

Display tibble

We can chek tibble with some method like following,

- glimpse: Display in a horizontal tabular format. All columns are visible.
- View: Display like pandas dataframe.
- Head: Display the first few lines of the View.
- kable: Display simple table that is not HTML.

```
glimpse(train)
```

```
## Rows: 891
## Columns: 12
## $ PassengerId <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,…
## $ Survived    <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1…
## $ Pclass      <dbl> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3…
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl…
## $ Sex         <chr> "male", "female", "female", "female", "male", "male", "mal…
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, …
## $ SibSp       <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0…
## $ Parch       <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0…
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37…
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625,…
## $ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G6", "C…
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"…
```

```
view(train)
head(train)
```

PassengerId <dbl>	Survived <dbl>	Pclass <dbl>	Name <chr>	Sex <chr>	A.. <dbl>	SibSp <dbl>
1	0	3	Braund, Mr. Owen Harris	male	22	1
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1
3	1	3	Heikkinen, Miss. Laina	female	26	0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1
5	0	3	Allen, Mr. William Henry	male	35	0
6	0	3	Moran, Mr. James	male	*NA*	0

6 rows | 1-7 of 12 columns

```
kable(train)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.00	1	0	A/5 21171	7.2500	NA	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38.00	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.00	0	0	STON/O2. 3101282	7.9250	NA	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.00	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.00	0	0	373450	8.0500	NA	S
6	0	3	Moran, Mr. James	male	NA	0	0	330877	8.4583	NA	Q
7	0	1	McCarthy, Mr. Timothy J	male	54.00	0	0	17463	51.8625	E46	S
8	0	3	Palsson, Master. Gosta Leonard	male	2.00	3	1	349909	21.0750	NA	S
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.00	0	2	347742	11.1333	NA	S
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.00	1	0	237736	30.0708	NA	C
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.00	1	1	PP 9549	16.7000	G6	S
12	1	1	Bonnell, Miss. Elizabeth	female	58.00	0	0	113783	26.5500	C103	S
13	0	3	Saundercock, Mr. William Henry	male	20.00	0	0	A/5. 2151	8.0500	NA	S
14	0	3	Andersson, Mr. Anders Johan	male	39.00	1	5	347082	31.2750	NA	S
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.00	0	0	350406	7.8542	NA	S
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.00	0	0	248706	16.0000	NA	S
17	0	3	Rice, Master. Eugene	male	2.00	4	1	382652	29.1250	NA	Q
18	1	2	Williams, Mr. Charles Eugene	male	NA	0	0	244373	13.0000	NA	S
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)	female	31.00	1	0	345763	18.0000	NA	S
20	1	3	Masselmani, Mrs. Fatima	female	NA	0	0	2649	7.2250	NA	C
21	0	2	Fynney, Mr. Joseph J	male	35.00	0	0	239865	26.0000	NA	S
22	1	2	Beesley, Mr. Lawrence	male	34.00	0	0	248698	13.0000	D56	S
23	1	3	McGowan, Miss. Anna	female	15.00	0	0	330923	8.0292	NA	Q

Check statistics

We can see statistics of dataframe by summary function. This is like describe method of pandas dataframe.

```
summary(train)
```

```
##   PassengerId       Survived          Pclass          Name
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000   Length:891
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000   Class :character
##  Median :446.0   Median :0.0000   Median :3.000   Mode  :character
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##      Sex                 Age            SibSp           Parch
##  Length:891         Min.   : 0.42   Min.   :0.000   Min.   :0.0000
##  Class :character   1st Qu.:20.12   1st Qu.:0.000   1st Qu.:0.0000
##  Mode  :character   Median :28.00   Median :0.000   Median :0.0000
##                     Mean   :29.70   Mean   :0.523   Mean   :0.3816
##                     3rd Qu.:38.00   3rd Qu.:1.000   3rd Qu.:0.0000
##                     Max.   :80.00   Max.   :8.000   Max.   :6.0000
##                     NA's   :177
##     Ticket               Fare           Cabin             Embarked
##  Length:891         Min.   :  0.00   Length:891         Length:891
##  Class :character   1st Qu.:  7.91   Class :character   Class :character
##  Mode  :character   Median : 14.45   Mode  :character   Mode  :character
##                     Mean   : 32.20
##                     3rd Qu.: 31.00
##                     Max.   :512.33
##
```

```
colSums(is.na(train))
```

```
## PassengerId    Survived      Pclass        Name         Sex         Age
##           0           0           0           0           0         177
##       SibSp       Parch      Ticket        Fare       Cabin    Embarked
##           0           0           0           0         687           2
```

Visualize the plot using ggplot

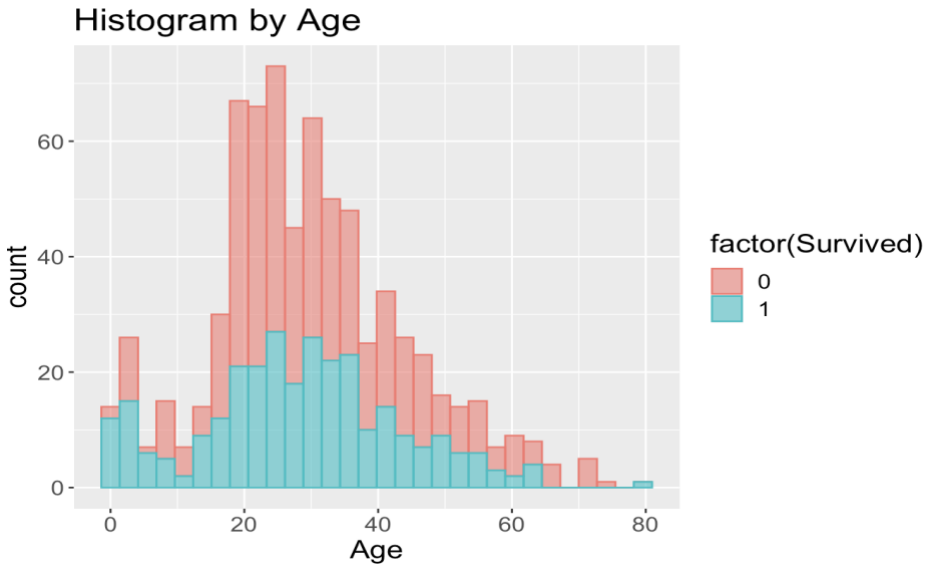
If we want to create plots in R, it is a good idea to start with [ggplot2]. If I'm forced to say, ggplot2 is similar to seaborn in python.

- In step1, we select data to use and define aesthetic relationship of columns of the data. We can do this by ggplot() and returns ggplot object.
- In step2, we choose geom_ function as plot type, and combine it to ggplot object.
- In step3, by combining more options into the ggplot object, we can customize it. For example, the axis format, title, and many other things

```
options(repr.plot.width=12, repr.plot.height=8)
```

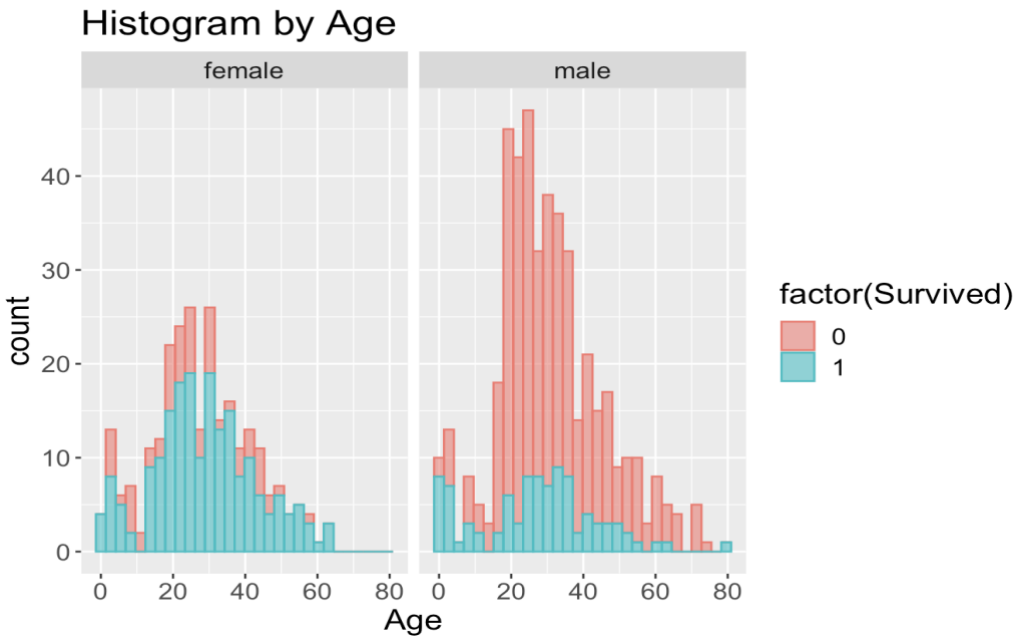
We can plot histogram by geom_histogram().

```
p <- ggplot(data = train, mapping = aes(x = Age, fill = factor(Survived), color = factor(Survived)))
p + geom_histogram(alpha = 0.6) + ggtitle("Histogram by Age") +
theme(text=element_text(size = 16))
```



The ratio of deaths seems to vary with age. Are there any other similar differences? For example, we are curious about gender.
By facet_wrap(), we can depicted graphs separately for each conditions very easily.

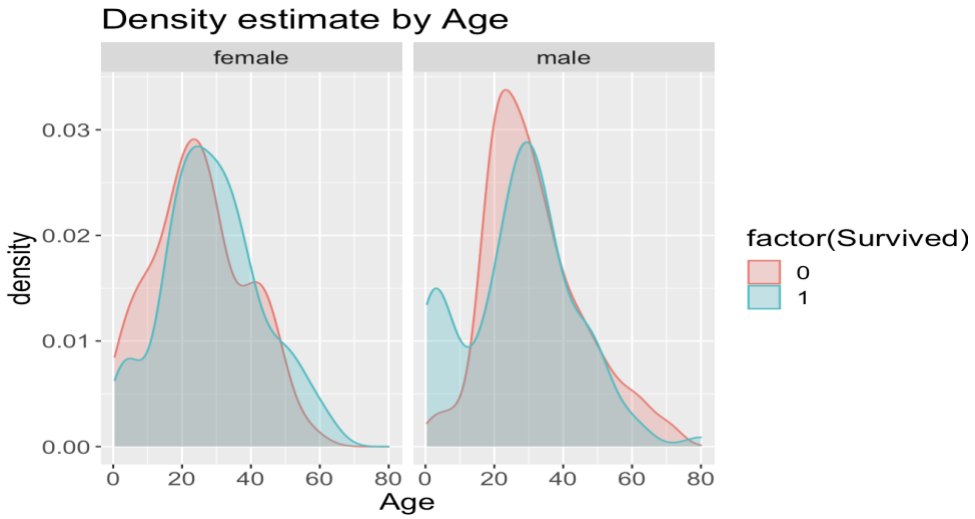
```
p <- ggplot(data = train, mapping = aes(x = Age, fill = factor(Survived), color = factor(Survived)))
p + geom_histogram(alpha = 0.6) + ggtitle("Histogram by Age") +
theme(text=element_text(size = 16)) + facet_wrap( ~ Sex, ncol = 2)
```



We can see that this difference is more pronounced for male.
We stratificated the data, now let's try to make the effect of age easier to understand by using geom_density() to draw the density.
We can plot the kernel density estimate by geom_density().

```
p <- ggplot(data = train, mapping = aes(x = Age, fill = factor(Survived), color = factor(Survived)))
p + geom_density(alpha = 0.3) + facet_wrap( ~ Sex, ncol = 2) +
```

```
theme(text=element_text(size = 16))  + ggtitle("Density estimate by Age")
```



If we look at the blue distribution of male, we can see that children have a higher survival rate because another peek appear at the age of below 20.
Now, since whether the person is child or not should have a lot to do with survival rate, let's draw a graph divided by that condition.
With functions of dplyr, we can manipurate this.
The definition of is_child is,
Yes: 1
No: 0

```
train <-  train %>% mutate(is_child = ifelse(Age < 18, 1, 0))
test <-  test %>% mutate(is_child = ifelse(Age < 18, 1, 0))
```

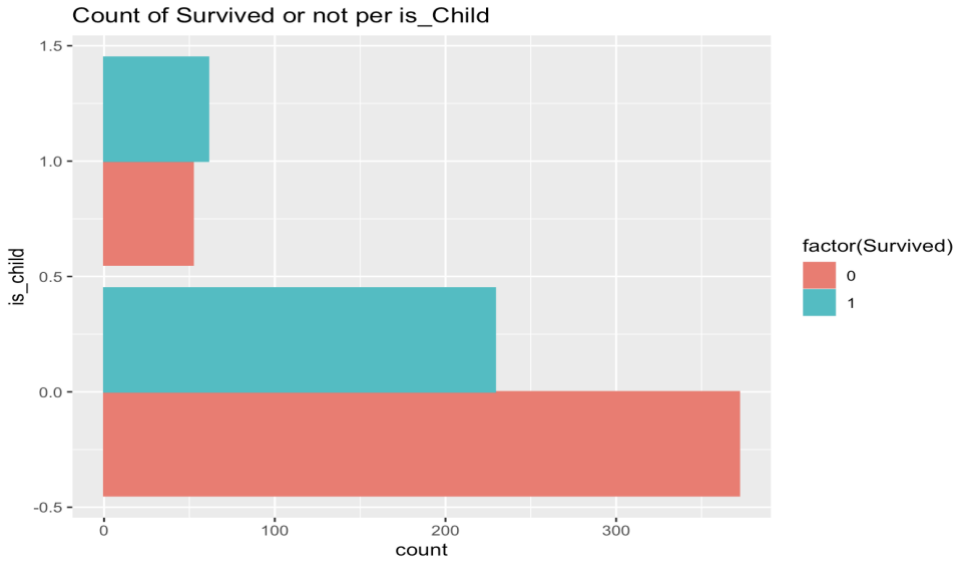
dplyr is a set of verbs that help you solve the most common data manipulation challenges. Data rangling can be done by connecting tibble and verv with "Pipe" (%>%). About feature engineering like this, I'll try more in later.
We can also do groupby and count by dplyr. It might be hard to get used to seeing at first, but we can makes it easier to see the whole operation since we can connect the operations behind.

```
Survived_count <- train %>%
  group_by(Survived, is_child) %>%
  summarize(count = n())
```

Draw a bar graph to see the impact of child or not more directly. The bar chart is geom_col().
If the condition is not category but continuous value such as real number (of course, if the number is distributed within a reasonable range of common sense), we can use factor().
We can also use coord_flip() if we want to make the plot horizontal. Anyway, if we combined coord_flip(), we can get the plot 90 degrees. In the case of Seaborn, the interface is little confusing. For barplot, the column names for the x and y arguments

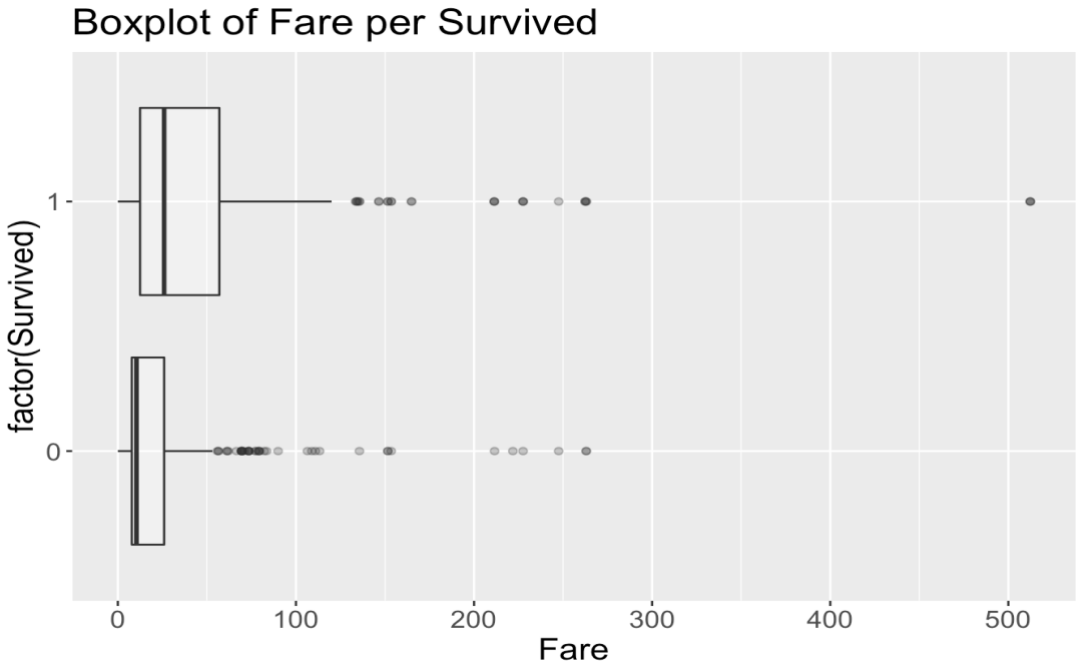
must be swapped, but for barplot, we have to pass 'h' to orient argment.
By plotting horizontally, we don't have to tilt the label names by 45 degrees, even if they are longer.

```
p <- ggplot(data = Survived_count, mapping = aes(x = is_child, y = count, fill = factor(Survived), color = factor(Survived)))
#By position_dodge(preserve = "single"), we can maintain width of bar.
p + geom_col(position = position_dodge(preserve = "single")) +
coord_flip() + ggtitle("Count of Survived or not per is_Child")
```



If they are children, more than half of them survive, but if they are not children, the survival rate is much lower.
We'll try to find other information that might have an impact on survival. If the fare are high, person may be prioritized for help.
Let's draw a boxplot to see if there is a difference in fare depending on whether you survived or not. This is geom_boxplot().

```
p <- ggplot(data = train, mapping = aes(x = factor(Survived), y = Fare))

p + geom_boxplot(alpha = 0.3) +

theme(text=element_text(size = 16)) +

coord_flip() + ggtitle("Boxplot of Fare per Survived")
```



As we can see from the boxplot, there is a difference in fare between groups which is whether we survived or not.
Let's calculate the statistics.

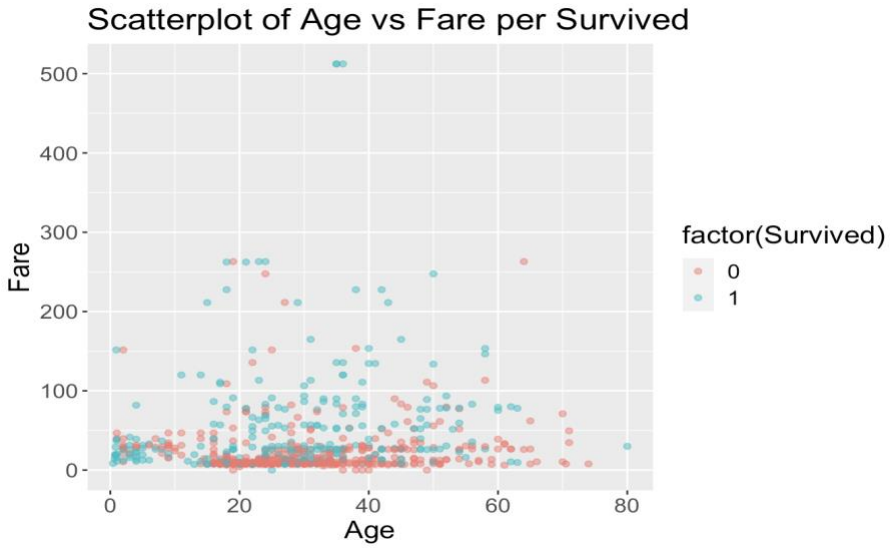
```
summary_fare <- train %>%
  group_by(Survived) %>%
  summarize(mean = mean(Fare), std_dev = sd(Fare))


summary_fare
```

Survived <dbl>	mean <dbl>	std_dev <dbl>
0	22.11789	31.38821
1	48.39541	66.59700

2 rows

The averages are about twice as different.
Let's delve further. Let's draw a scatterplot of each data for each age. We can plot scatterplot by geom_point().

```
p <- ggplot(data = train, mapping = aes(x = Age, y = Fare, color = factor(Survived)))
p + geom_point(alpha = 0.5) + theme(text=element_text(size = 16)) +
ggtitle("Scatterplot of Age vs Fare per Survived")
```

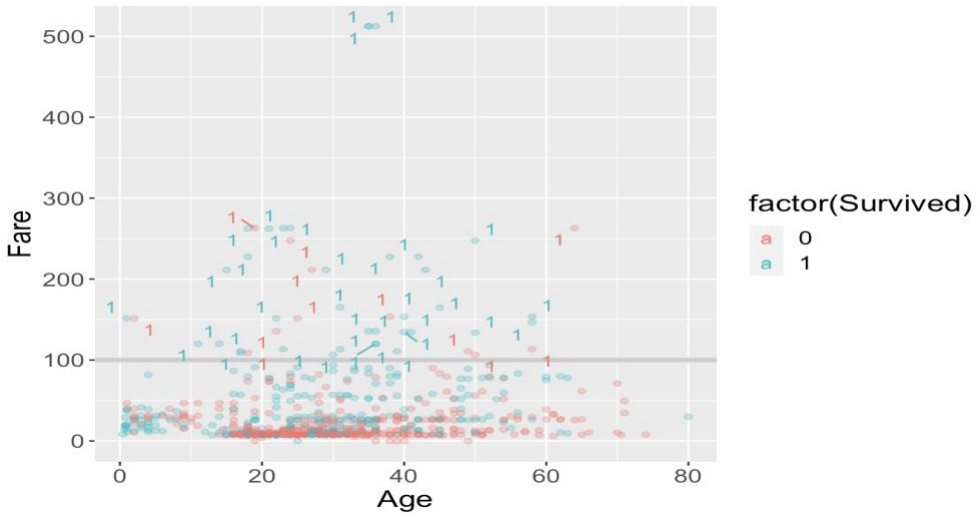


In particular, in geom_point(), by specifying **a**, the color becomes deeper where the number of data is high, which is good for visual clarity.
We can see that the higher the Fare, the more green dots there are. Also, as we saw above, the green dots are concentrated where the age is low.
There is one more column that is related to Fare. It is the Ticket class.

- 1st: Upper
- 2nd: Middle
- 3rd: Lower For points with Fare higher than 100, let's see which class they are by assigning labels of pclass. Also, let's draw a horizontal line at 100 to make the boundary easier to understand.

We can overlap labels to points by geom_text_repel() of ggrepel. And we can overwrite hline by geom_hline().

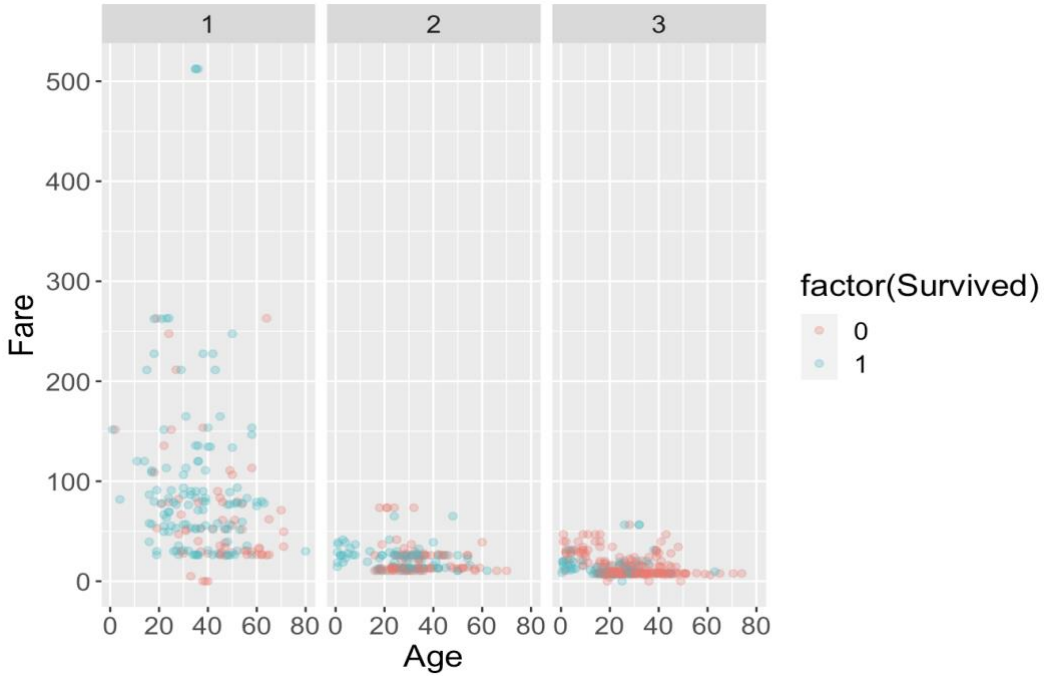
```
p <- ggplot(data = train, mapping = aes(x = Age, y = Fare, color = factor(Survived)))
p + geom_point(alpha = 0.3) + theme(text=element_text(size = 16)) +
geom_hline(yintercept = 100, size = 1.4, color = "gray80") +
geom_text_repel(data = subset(train, Fare > 100), mapping = aes(label = Pclass))
```



As a matter of course, we can see that all data with a Fare greater than 100 have a pclass

of 1. It also shows a high percentage of survivors.
If we look at the graphs separately by pclass, we may be able to see more features.

```
p <- ggplot(data = train, mapping = aes(x = Age, y = Fare, color = factor(Survived)))
p + geom_point(alpha = 0.3) + theme(text=element_text(size = 16)) +
facet_wrap(~ Pclass)
```



We can see that the percentage of survivors is higher for smaller values of pclass. pclass 3 has more red dots. Children are mostly green.
So far, we have revealed the following features.

- Children seem to have a better chance of survival.
- Female seem to have a higher survival rate than male.
- The higher fare or ticket class, the higher the survival rate seems to be.

Feature Engineering

To more accuracy, I'll try more feature engineering.
I referred A Data Science Framework: To Achieve 99% Accuracy for the features to be created.

Concat data to process & select sub-set

```
target_cols <- c("Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked")
# Combine the train data and test data for processing.
train_and_test <- rbind(train[, target_cols], test[, target_cols])
# Extract target
```

```
train_label <- train$Survived
```

Fill missing values with statistics

```
# https://stackoverflow.com/questions/2547402/how-to-find-the-statistical-mode
mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy
train_and_test$Age[is.na(train_and_test$Age)]<-median(train_and_test$Age,na.rm=TRUE)
train_and_test$Embarked[is.na(train_and_test$Embarked)]<-mode(train_and_test$Embarked)
train_and_test$Fare[is.na(train_and_test$Fare)]<-median(train_and_test$Fare,na.rm=TRUE)
```

Categorical encoding

```
train_and_test$Sex <- as.numeric(factor(train_and_test$Sex))

train_and_test$Embarked <- as.numeric(factor(train_and_test$Embarked))
```

Create new column from some columns

```
train_and_test$FamilySize = train_and_test$SibSp + train$Parch + 1
train_and_test = train_and_test %>% mutate(IsAlone = ifelse(FamilySize > 1, 1, 0))
```

Binning & label encoding

```
train_and_test$Age_bin <- bin(train_and_test$Age, nbins = 5, labels = c(0, 1, 2, 3, 4))
train_and_test <- train_and_test %>% mutate(Age_bin = as.integer(Age_bin))

train_and_test$Fare_bin <- bin(train_and_test$Fare, nbins = 5, labels = c(0, 1, 2, 3, 4))
train_and_test <- train_and_test %>% mutate(Fare_bin = as.integer(Fare_bin))
```

Horizontal division of tibble

```
train_idx <- 1:nrow(train)
test_idx <- -train_idx
```

```
train <- train_and_test[train_idx,]
test <- train_and_test[test_idx,]
```

Finally, we have our data set!

```
input_cols <- append(target_cols, c( "FamilySize", "IsAlone", "Age_bin", "Fare_bin"))
train_data <- train[, input_cols]
test_data <- test[, input_cols]
```

Modeling with LightGBM

As Python users, you probably want to try LightGBM first after EDA. This method is also available in R. Let's create a model and submit the estimation results.

Create dataset

In order to pass the tibble to the LightGBM Dataset API, we have to made it into matrix using as.matrix().

```
cat_cols <- c("Pclass", "Sex", "Embarked", "IsAlone", "Age_bin", "Fare_bin")
train_ds <- lgb.Dataset(as.matrix(train_data), label = as.matrix(train_label), categorical_feature = cat_cols)
```

Cross validation

```
lgb.cv(
    objective = "binary"
  , metric = "binary_logloss"
  , data = train_ds
  , nrounds = 1500L
  , learning_rate = 0.002
  , early_stopping_rounds = 10
  , nfold = 5
  , shuffle = TRUE
)
```

```
## <lgb.CVBooster>
##   Public:
##     best_iter: 1257
##     best_score: 0.412052887156094
##     boosters: list
##     initialize: function (x)
##     record_evals: list
##     reset_parameter: function (new_params)
```

Training

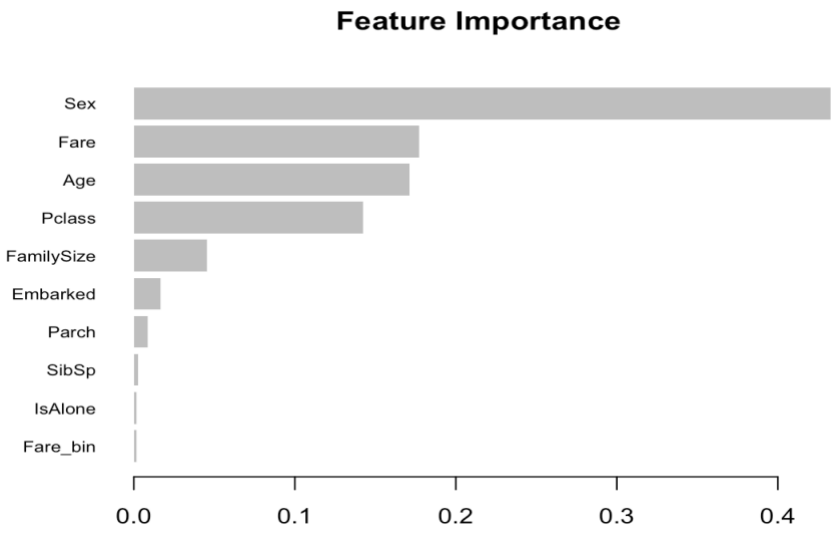
```
model <- lightgbm(
   objective = "binary"
   , metric = "binary_logloss"
   , data = train_ds
   , nrounds = 1433
   , learning_rate = 0.002
)
```

```
## [1] "[1425]:  train's binary_logloss:0.300965"
## [1] "[1426]:  train's binary_logloss:0.300897"
## [1] "[1427]:  train's binary_logloss:0.300831"
## [1] "[1428]:  train's binary_logloss:0.300763"
## [1] "[1429]:  train's binary_logloss:0.300696"
## [1] "[1430]:  train's binary_logloss:0.300629"
## [1] "[1431]:  train's binary_logloss:0.300563"
## [1] "[1432]:  train's binary_logloss:0.300496"
## [1] "[1433]:  train's binary_logloss:0.300431"
```

Check feature importance

```
tree_imp <- lgb.importance(model, percentage = TRUE)
```

```
lgb.plot.importance(
  tree_imp,
  top_n = 10L,
  measure = "Gain",
  left_margin = 10L,
  cex = NULL
)
```



Inference & Export Submission File

```
preds <- predict(model, as.matrix(test_data))
sub$Survived <- ifelse(preds > 0.45,1,0)
write.csv(sub, "titanic/submission.csv", row.names=FALSE)
```

GitHub Link of Project: https://bit.ly/3awmbZP