Using an OLS Model to Predict COVID-19 Patient Length of Stay

B204976

2024-12-09

```
library(dplyr)
library(sparklyr)
library(ggplot2)
library(knitr)
library(kableExtra)
library(corrr)
library(janitor)
library(dbplot)
library(broom)
```

```
sc <- spark_connect(master = "local")
covid_sprk <- spark_read_csv(sc, name = "covid_sprk", path = "data/host_train.csv")
```

Introduction

This report utilizes the data set from 'COVID-19 Hospitals Treatment Plan', and intends to predict length of stay (LoS) from other independent variables contained in the dataset using an Ordinary Least Squares (OLS) Regression model.

```
glimpse(covid_sprk)
```

```
## Rows: ??
## Columns: 18
## Database: spark_connection
## $ case_id                           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 1~
## $ Hospital                          <int> 8, 2, 10, 26, 26, 23, 32, 23, 1, 10,~
## $ Hospital_type                     <int> 2, 2, 4, 1, 1, 0, 5, 0, 3, 4, 6, 1, ~
## $ Hospital_city                     <int> 3, 5, 1, 2, 2, 6, 9, 6, 10, 1, 9, 2,~
## $ Hospital_region                   <int> 2, 2, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, ~
## $ Available_Extra_Rooms_in_Hospital <int> 3, 2, 2, 2, 2, 2, 1, 4, 2, 2, 2, 4, ~
## $ Department                        <chr> "radiotherapy", "radiotherapy", "ane~
## $ Ward_Type                         <chr> "R", "S", "S", "R", "S", "S", "S", "~
## $ Ward_Facility                     <chr> "F", "F", "E", "D", "D", "F", "B", "~
## $ Bed_Grade                         <dbl> 2, 2, 2, 2, 2, 2, 3, 3, 4, 3, 2, 1, ~
## $ patientid                         <int> 31397, 31397, 31397, 31397, 31397, 3~
## $ City_Code_Patient                 <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, ~
## $ Type_of_Admission                 <chr> "Emergency", "Trauma", "Trauma", "Tr~
## $ Illness_Severity                  <chr> "Extreme", "Extreme", "Extreme", "Ex~
## $ Patient_Visitors                  <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
## $ Age                               <chr> "51-60", "51-60", "51-60", "51-60", ~
```

```
## $ Admission_Deposit                 <dbl> 4911, 5954, 4745, 7272, 5558, 4449, ~
## $ Stay_Days                         <chr> "0-10", "41-50", "31-40", "41-50", "~
```

The independent variables I have selected are `Type_of_Admission`, `Illness_Severity`, `Bed_Grade`, `Patient_Visitors` and `Available_Extra_Rooms_in_Hospital`. The dependent variable is `Stay_Days`.

Exploratory Data Analysis & Transformation

The first step is to split the data into a training and testing set, by limiting my analysis to the training set I will be able to avoid data leakage and bias in the model.

```
data_splits <- covid_sprk %>%
  sdf_random_split(training = 0.8, testing = 0.2, seed = 68)
#seed means that the same split is done each time the code is run
covid_train <- data_splits$training
covid_test <- data_splits$testing
```

Then, I created a function that will allow me to see each kind of value in a column.

```
distinct_values <- function(column_name) {
  covid_train %>%
    select(all_of(column_name)) %>% #select the entire column of interest
    distinct() %>% #each different kind of value.
    print(n = 40) #so that all values are printed
}
```

The dependent variable, `Stay_Days`, is a character. This must be converted to a numerical variable.

```
distinct_values("Stay_Days")
```

```
## # Source:   SQL [11 x 1]
## # Database: spark_connection
##    Stay_Days
##    <chr>
##  1 0-10
##  2 71-80
##  3 11-20
##  4 More than 100 Days
##  5 31-40
##  6 41-50
##  7 21-30
##  8 91-100
##  9 51-60
## 10 61-70
## 11 81-90
```

The dependent variable, Stay_Days, is a given a series of integer day ranges in the data, that I am proposing to convert to a single numerical variable for subsequent analysis, encoding each range at its midpoint e.g. '0 to 10 days' as 5 days.

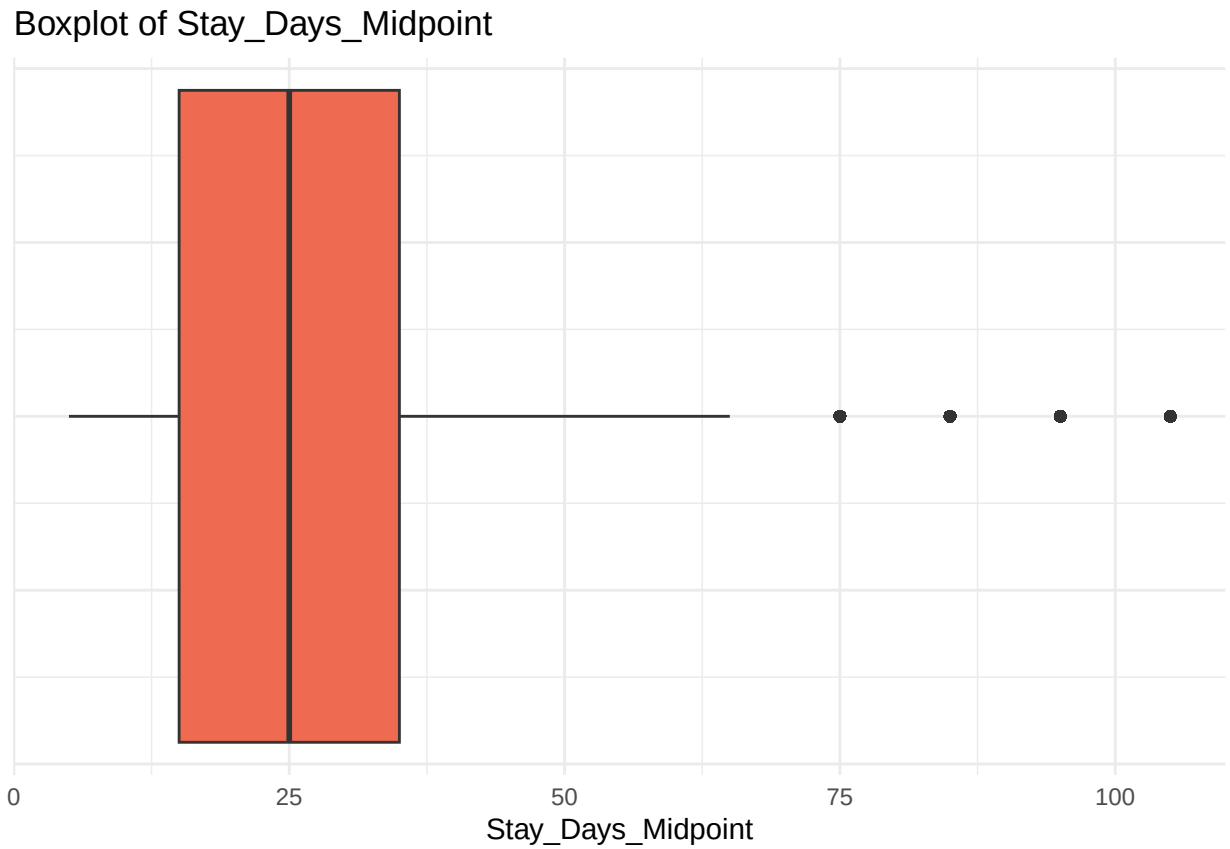
Midpoint is not meaningful for the "more than 100 days", so to determine whether this sub-group is a material part of the dataset I conducted the following analysis:

```r
more_than_100_count <- covid_train %>%
  filter(Stay_Days == "More than 100 Days") %>%
  summarise(count = n()) %>% #count the number of times this value appears
  collect() %>% #convert to R object, from spark.
  pull(count) #pull just the number.

total_count <-covid_train %>%
  summarise(count = n()) %>%
  collect() %>%
  pull(count)

percent_more_than_100_days <- (more_than_100_count / total_count * 100)
#calculate the % of patients who stayed more that 100 days
print(percent_more_than_100_days)
```

```
## [1] 2.09948
```

"More than 100 Days" makes up 2% of patients. This is a small group, so I have determined that it is acceptable to encode it as 105. Then I created a boxplot to see the distribution of Stay_Days_Midpoint.

```r
covid_train <- covid_train %>%
  mutate(Stay_Days_Midpoint = case_when(
      Stay_Days == "0-10" ~ 5,
      Stay_Days == "11-20" ~ 15,
      Stay_Days == "21-30" ~ 25,
      Stay_Days == "31-40" ~ 35,
      Stay_Days == "41-50" ~ 45,
      Stay_Days == "51-60" ~ 55,
      Stay_Days == "61-70" ~ 65,
      Stay_Days == "71-80" ~ 75,
      Stay_Days == "81-90" ~ 85,
      Stay_Days == "91-100" ~ 95,
      Stay_Days == "More than 100 Days" ~ 105))

covid_train %>%
  sdf_sample(fraction = 0.1, replace = TRUE, seed = 11) %>%
  select(Stay_Days_Midpoint) %>%
  collect() %>%
  ggplot(aes(x = Stay_Days_Midpoint)) +
  geom_boxplot(fill = "coral2") +
  theme_minimal() +
  labs(title = "Boxplot of Stay_Days_Midpoint") +
  theme(axis.text.y = element_blank(), axis.ticks.y = element_blank())
```



50% of the patients in this sample satyed between 15 and 35 days, and the distribution does not appear to be symmetric, as the right whisker is longer than the left, and there 4 outliers greater than 1.5 times the IQR. As `Stay_Days_Midpoint` can only be positive, I decided to log it to create an OLS model where the dependent variable can never be negative.

```
covid_train <- covid_train %>%
  mutate(ln_Stay_Days_Midpoint = log(Stay_Days_Midpoint))
#create a column containing the log of Stay_Days_Midpoint
```

Then, I one-hot-encoded the categorical variable `Type_of_Admission`.

```
distinct_values("Type_of_Admission")
```

```
## # Source:   SQL [3 x 1]
## # Database: spark_connection
##   Type_of_Admission
##   <chr>
## 1 Emergency
## 2 Urgent
## 3 Trauma
```

```
covid_train <- mutate(covid_train,
  Emergency = ifelse(Type_of_Admission == "Emergency", 1, 0),
  Urgent = ifelse(Type_of_Admission == "Urgent", 1, 0),
  Trauma = ifelse(Type_of_Admission == "Trauma", 1, 0)
)
```

I did ordinal encoding on `Illness_Severity`, because there is an inherent order in severity, therefore one-hot-encoding may not be suitable.

```
distinct_values("Illness_Severity")
```

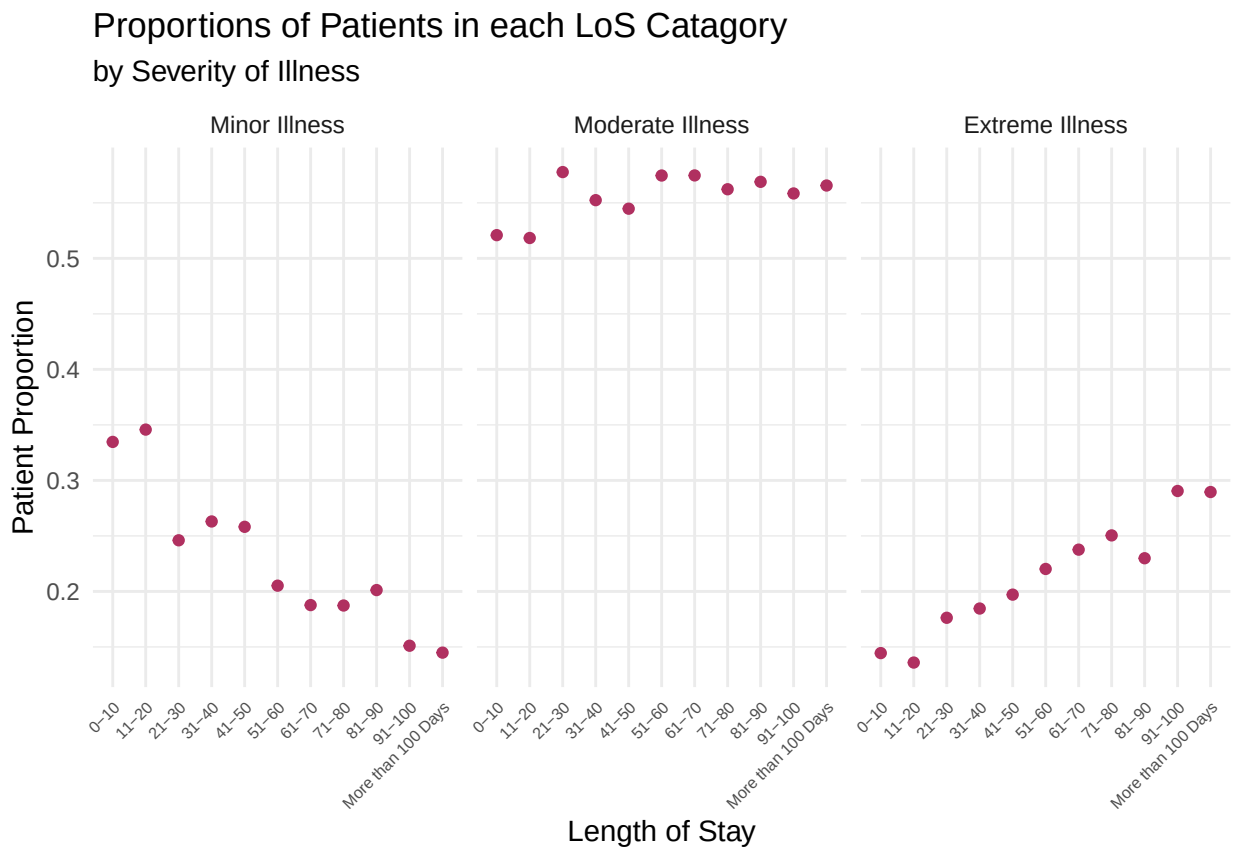
```
## # Source:   SQL [3 x 1]
## # Database: spark_connection
##   Illness_Severity
##   <chr>
## 1 Extreme
## 2 Minor
## 3 Moderate
```

```
covid_train <- covid_train %>%
  mutate(Illness_Severity = case_when(
  Illness_Severity == "Minor" ~ 1,
  Illness_Severity == "Moderate" ~ 2,
  Illness_Severity == "Extreme" ~ 3)
)
```

Next, I compared `Stay_Days` to `Illness_Severity` to see if there is any correlation between illness-severity and LoS.

```
contingency_tbl_IS <- covid_train %>%
  sdf_crosstab("Stay_Days", "Illness_Severity") %>%
  collect() %>%
  rename("Minor" = 2,
         "Moderate" = 3,
         "Extreme" = 4) %>%
#Rename the values in cols 2, 3 and 4 back to severity for ease of reading
#the graph.
  arrange(Stay_Days_Illness_Severity)
#Arrange in ascending order of LoS.

contingency_tbl_IS <- contingency_tbl_IS %>%
  group_by(Stay_Days_Illness_Severity) %>%
   mutate(total = Minor + Moderate + Extreme) %>%
#Create col with the totals
  mutate(
    "Minor Illness" = Minor / total,
    "Moderate Illness" = Moderate / total,
    "Extreme Illness" = Extreme / total
  ) %>%
   #Calculate the proportion for each illness severity category
  select(-c(2:4)) %>%
  select(-total)
 #Remove the unnecessary columns (not containing proportions)

contingency_tbl_IS %>%
  pivot_longer(cols = c(2:4),
               names_to = "Severity",
               values_to = "Count") %>%
  #pivot_longer so that graph can be facet_wrapped
```

```
  mutate(Severity = factor(Severity, levels = c("Minor Illness",
                                                "Moderate Illness",
                                                "Extreme Illness")) )%>%
#Assign levels to Severity, so they appear in order on the graph
ggplot(aes(x = Stay_Days_Illness_Severity,
           y = Count))+
geom_point(colour = "maroon") +
facet_wrap(~Severity) +
theme_minimal() +
theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 6))+
labs(x = "Length of Stay",
     y = "Patient Proportion",
     title = "Proportions of Patients in each LoS Catagory",
     subtitle = "by Severity of Illness")
```



This graph demonstrate that most patients with minor disease stayed for shorter periods. Patients with moderate illness has similar rates across all LoS. Most patients with severe illness stayed for longer periods. Therefore, there is some correlaiton between severity and LoS.
Then I included `Bed_Grade`, because the condition of the bed in the ward might impact LoS.

```
distinct_values("Bed_Grade") #NA values present
```

```
## # Source:   SQL [5 x 1]
## # Database: spark_connection
##   Bed_Grade
##       <dbl>
```

```
## 1         4
## 2         1
## 3        NA
## 4         3
## 5         2
```

```
covid_train %>%
  summarise(mean = mean(Bed_Grade, na.rm = TRUE)) %>%
  #calculate mean of Bed_Grade
  collect()
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  2.63
```

```
covid_train <- covid_train %>%
  mutate(Bed_Grade = case_when(
  is.na(Bed_Grade) ~ 2.63,
  Bed_Grade == "1" ~ 1,
  Bed_Grade == "2" ~ 2,
  Bed_Grade == "3" ~ 3,
  Bed_Grade == "4" ~ 4)
) #replacing NA with the mean of bed grade
```

Given that I decided to include `Patient_Visitors`, I looked at the correlation between `Patient_Visitors` and `Stay_Days_Midpoint`.

```
correlation_Visitor_Stay <-ml_corr(covid_train,
                                   columns = c("Patient_Visitors",
                                               "Stay_Days_Midpoint"))

kable(correlation_Visitor_Stay,
      col.names = c("Patient Visitors", "Stay Days Midpoint"),
        align = "cc",
        caption = "Correlation of Patient Visitors and Stay Days Midpoint",
        digits = 2)
```

Table 1: Correlation of Patient Visitors and Stay Days Midpoint

Patient Visitors	Stay Days Midpoint
1.00	0.54
0.54	1.00

The correlation value of 0.54 suggests moderate positive correlation between these two variables. `Available_Extra_Rooms_in_Hospital` is already numeric, so does not need to be encoded. I will, however, shorten the name of this variable, so that it better fits in tables.

```
covid_train <- covid_train %>%
  mutate(Available_Rooms = Available_Extra_Rooms_in_Hospital)
#shorten name for use in table later.
```

OLS Model.

An Ordinary Least Squares (OLS) model is used to estimate the relationship between independent variables and a dependent variable by minimizing the sum of the squared differences between the observed and predicted values.

```
ols_model <- ml_linear_regression(covid_train,
                                  ln_Stay_Days_Midpoint ~
                                    Illness_Severity +
                                    Bed_Grade +
                                    Urgent +
                                    Emergency +
                                    Patient_Visitors +
                                    Available_Rooms)
```

```
model_results <- tidy(ols_model) %>%
  mutate(
    # Calculate the lower and upper 95% CI
    lower = round(estimate - 1.96 * std.error, 2),
    upper = round(estimate + 1.96 * std.error, 2),
    # Round the estimate to 2 decimal places
    estimate = round(estimate, 2),
    # Create a new column with the estimate and CI
    estimate_95_ci = paste(estimate, "[", lower, ",", upper, "]")
  )

model_results %>%
  select(term, p.value, estimate_95_ci) %>%
kable(
      col.names = c("Term", "p-value", "Estimate [ 95% CI]"),
        align = "lcc",
        caption = "Model Results",
        digits = 2)
```

Table 2: Model Results

Term	p-value	Estimate [95% CI]
(Intercept)	0	2.97 [2.96 , 2.99]
Illness_Severity	0	0.08 [0.08 , 0.08]
Bed_Grade	0	-0.04 [-0.04 , -0.04]
Urgent	0	-0.1 [-0.11 , -0.09]
Emergency	0	-0.24 [-0.25 , -0.23]
Patient_Visitors	0	0.18 [0.18 , 0.18]
Available_Rooms	0	-0.08 [-0.08 , -0.08]

The estimate for `Illness_Severity` is 0.08. So there is a small, positive effect of `Illness_Severity` on the dependent variable. `Bed_Grade`, and `Urgent` both have small and negative estimates (-0.04, and -0.10), so have small, negative impacts on the dependent variable. The `Emergency` variable has a greater negative impact on the dependent variable with an estimate of -0.24. `Patient_Visitors` has an estimate of 0.18 and therefore has a small, positive impact on the dependent variable. Finally, `Available_Extra_Rooms_in_Hospital` has a very small impact on the dependent variable, with a negative estimate of -0.08.

All of the variables in this model have very narrow confidence intervals (CIs) and very small, statistically significant p-values, likely due to the large size of the dataset.

I then applied the data transformation to `covid_test`, and used this to evaluate the model.

```
covid_test <- covid_test %>%
  mutate(Stay_Days_Midpoint = case_when(
      Stay_Days == "0-10" ~ 5,
      Stay_Days == "11-20" ~ 15,
      Stay_Days == "21-30" ~ 25,
      Stay_Days == "31-40" ~ 35,
      Stay_Days == "41-50" ~ 45,
      Stay_Days == "51-60" ~ 55,
      Stay_Days == "61-70" ~ 65,
      Stay_Days == "71-80" ~ 75,
      Stay_Days == "81-90" ~ 85,
      Stay_Days == "91-100" ~ 95,
      Stay_Days == "More than 100 Days" ~ 105))

covid_test <- covid_test %>%
  mutate(ln_Stay_Days_Midpoint = log(Stay_Days_Midpoint))
```

```
covid_test <- covid_test %>%
  mutate(Illness_Severity = case_when(
  Illness_Severity == "Minor" ~ 1,
  Illness_Severity == "Moderate" ~ 2,
  Illness_Severity == "Extreme" ~ 3)
)
```

```
covid_test <- mutate(covid_test,
  Emergency = ifelse(Type_of_Admission == "Emergency", 1, 0),
  Urgent = ifelse(Type_of_Admission == "Urgent", 1, 0),
  Trauma = ifelse(Type_of_Admission == "Trauma", 1, 0)
)
```

```
covid_test <- covid_test %>%
  mutate(Bed_Grade = case_when(
  Bed_Grade == "1" ~ 1,
  Bed_Grade == "2" ~ 2,
  Bed_Grade == "3" ~ 3,
  Bed_Grade == "4" ~ 4,
  is.na(Bed_Grade) ~ 2.63)
  #replacing NA with the mean of bed grade from covid_train
)
```

```
covid_test <- covid_test %>%
  mutate(Available_Rooms = Available_Extra_Rooms_in_Hospital)
```

Model Evaluation

```r
model_metrics <- ml_evaluate(ols_model, covid_test)

metrics_table <- data.frame(
  Metric = c("R²", "Adjusted R²", "Mean Absolute Error (MAE)",
             "Mean Squared Error (MSE)", "Root Mean Squared Error (RMSE)"),
  Value = c(model_metrics$r2,
    model_metrics$r2adj,
    model_metrics$mean_absolute_error,
    model_metrics$mean_squared_error,
    model_metrics$root_mean_squared_error))

kable(metrics_table,
      align = "lc",
      caption = "Summary of Model Evaluation Metrics",
      digits = 2)
```

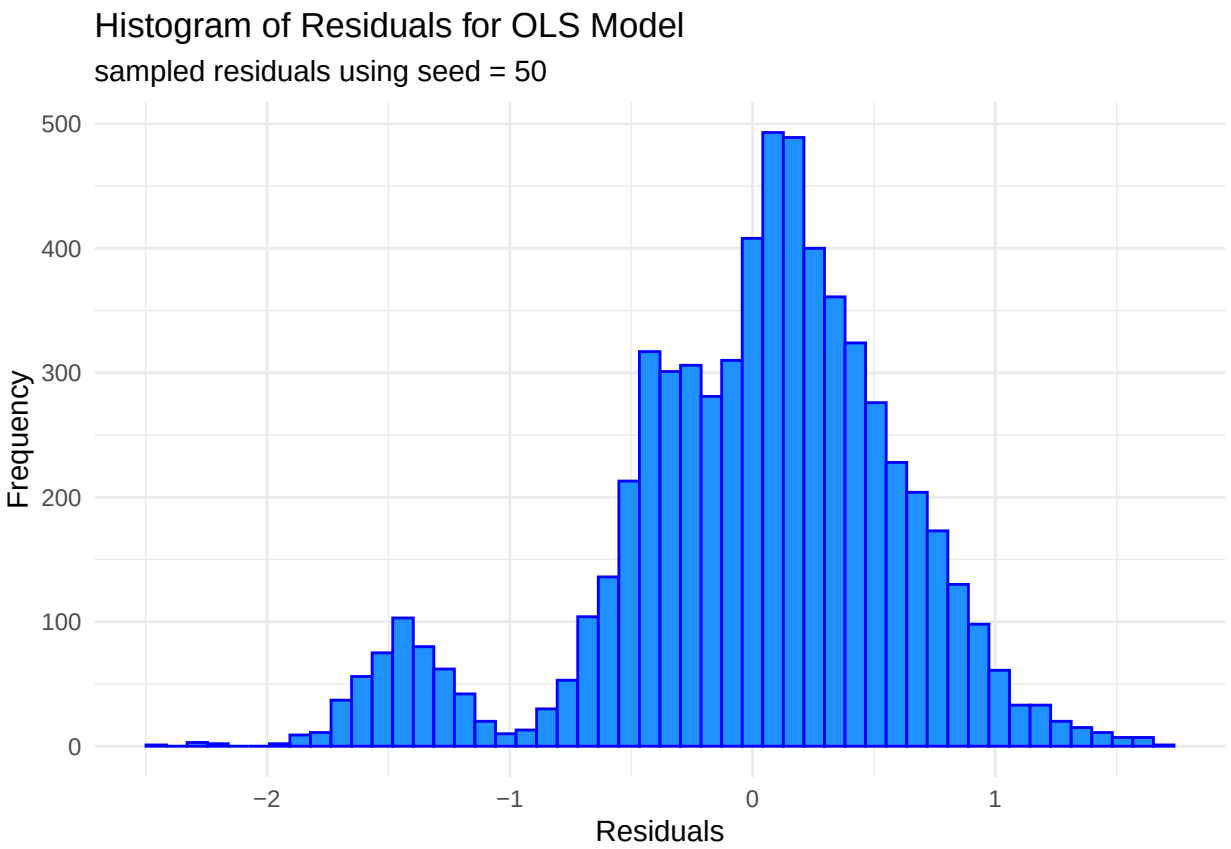
Table 3: Summary of Model Evaluation Metrics

Metric	Value
R^2	0.24
Adjusted R^2	0.24
Mean Absolute Error (MAE)	0.46
Mean Squared Error (MSE)	0.37
Root Mean Squared Error (RMSE)	0.61

An 0.24 R^2 value indicates that the model explains ~24% of the variance in the dependent variable, a relatively weak fit, meaning that other variables or nonlinear relationships might be influencing ln_Stay_Days_Midpoint. The adjusted R^2 is identical to R^2, suggesting that the number of predictors used is appropriate. The MAE of 0.46 means the model's predictions for log(Stay_Days_Midpoint) are off by about 0.46 log-scale units or 1.59 days.

A moderate MSE of 0.37 suggests that the model's predictions are off by a squared difference of 0.37 on the log scale, or 1.45 days. Squaring errors emphasizes larger more than smaller mistakes, demonstrating that the model works fairly well, occasionally making larger errors. An RMSE of 0.61 units on the log scale, or 1.84 days and provides an error measure in the same units as the target variable.

```r
residuals <- model_metrics$residuals()

residuals %>%
  sdf_sample(fraction = 0.1, replacement = TRUE, seed = 50) %>%
#sampling the 'residuals' spark dataframe, to plot residual distribution
  collect() %>%
  ggplot(aes(x = residuals)) +
  geom_histogram(bins = 50,
                 colour = "blue",
                 fill = "dodgerblue") +
  labs(title = "Histogram of Residuals for OLS Model",
       subtitle = "sampled residuals using seed = 50",
       x = "Residuals",
       y = "Frequency")+
  theme_minimal()
```



There is mostly normal distribution of the residuals, but there is some bimodal shape to the histogram, which suggests that linear regression may not be the perfect model for this data.

Conclusions

This OLS model is able to predict 24% of the variance in LoS however, a relatively low R^2 indicates that additional variables or more complex models could enhance predictive power and that there are other factors or non-linear relationships influencing LoS. Illness_Severity, Patient_Visitors, and Admission_Type showed small but statistically significant effects on LoS. Strengths in this method include the comprehensive transformation steps, along with the use of a large data set and the simplicity and interpretability of the model. However, there are weaknesses in the limited variance explanation and possible over-simplification of the relationships between variables, as evidenced by the distribution of the residuals. Further improvement could utilise more complex models to capture the relationships between the dependent and independent variables. It is also possible that the dataset itself does not contain adequate information to predict LoS, and that factors such as comorbidities, vaccination-status and medication could have greater impact. Overall, this analysis could inform hospital resource management but requires refinement for higher accuracy in real-world applications.

Word count: 973