University of Edinburgh Usher Institute
Data Skills Workforce Development
Data Science for Health and Social Care programme
Big data analytics (HEIN11055)

R programming assignment - an exploration of machine learning models for the prediction and classification of the length of hospital stay for Covid-19 patients, using patient data and the Sparklyr library

Exam number: B010574

09 December 2022

Contents

1 Overview

The supplied dataset shows patient history in a hospital, with features such as hospital department, type of admission, severity of illness and age. Although the data comes from Kaggle (https://www.kaggle.com/datasets/arashnic/covid19-hospital-treatment), the original source of the data is unclear and an assumption is made that the data is artificial/synthetic/anonymised.

The data is treated as 'big data' and the Sparklyr library is used for this.

The supplied dataset is in a well-structured 'tidy' CSV format and has a sufficiently rich range of features/predictor values (wide, 18) and records/observations (long, 318438) for a machine learning approach.

The data also has accompanying metadata on the Kaggle website, as well as a data dictionary and a helpful graphical interface showing the distribution and properties (summary statistics) of each feature in the dataset.

The set of features (or input, or independent variables) also includes labelled values for the 'Stay_Days' feature, so a supervised learning model and an evaluation of prediction accuracy is possible with this data, using 'Stay_Days' as a predicted output/target or dependent variable of a machine learning model.

The models used in this analysis will predict the classification (length of hospital stay) using the supplied data about a patient. The models will use a multinomial approach as the length of stay is a categorical feature with 11 classifications in the data.

2 Analysis

2.1 Libraries

```
library(sparklyr)
library(dplyr)
library(knitr)
library(dbplot)
library(formatR)
```

2.2 Spark connection and import data

```
# Create spark connection (local), import Kaggle data
sc <- spark_connect(master = "local", version = "2.3")
data <- spark_read_csv(sc, "host_train.csv")
```

2.3 Exploratory Data Analysis

2.3.1 Basic details

```
# Basic overview
count(data)
```

```
# Source: spark<?> [?? x 1]
       n
   <dbl>
1 318438
```

```
glimpse(data)
```

```
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", "~
```

```
# Function to transpose data frame and retain column names in rows
transpose_df <- function(df) {
    t_df <- data.table::transpose(df)
    colnames(t_df) <- rownames(df)
    rownames(t_df) <- colnames(df)
    t_df <- t_df %>%
        tibble::rownames_to_column(.data = .) %>%
        tibble::as_tibble(.)
    return(t_df)
}

# Totals of missing data in each feature (transpose the output data frame)
missing_count <- data %>%
    summarise_all(~sum(as.integer(is.na(.)))) %>%
    collect() %>%
    transpose_df %>%
    rename(Feature = 1, `No. missing` = 2) %>%
    print(n = 18)
```

```
# A tibble: 18 x 2
   Feature                          `No. missing`
   <chr>                                    <dbl>
 1 case_id                                      0
 2 Hospital                                     0
 3 Hospital_type                                0
 4 Hospital_city                                0
 5 Hospital_region                              0
 6 Available_Extra_Rooms_in_Hospital            0
```

```
 7 Department                                0
 8 Ward_Type                                 0
 9 Ward_Facility                             0
10 Bed_Grade                               113
11 patientid                                 0
12 City_Code_Patient                      4532
13 Type_of_Admission                         0
14 Illness_Severity                          0
15 Patient_Visitors                          0
16 Age                                       0
17 Admission_Deposit                         0
18 Stay_Days                                 0
```

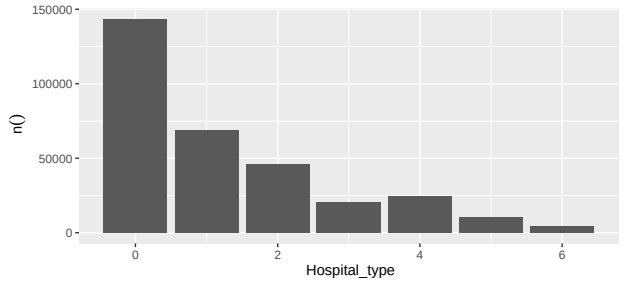
```
# Group by each target feature (Stay_Days) category and count
stay_days_count <- data %>%
    group_by(Stay_Days) %>%
    summarise(count = n()) %>%
    collect() %>%
    arrange(Stay_Days) %>%
    print(n = 11)
```

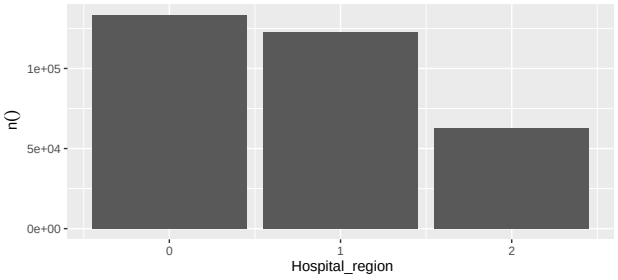
```
# A tibble: 11 x 2
   Stay_Days          count
   <chr>              <dbl>
 1 0-10               23604
 2 11-20              78139
 3 21-30              87491
 4 31-40              55159
 5 41-50              11743
 6 51-60              35018
 7 61-70               2744
 8 71-80              10254
 9 81-90               4838
10 91-100              2765
11 More than 100 Days  6683
```

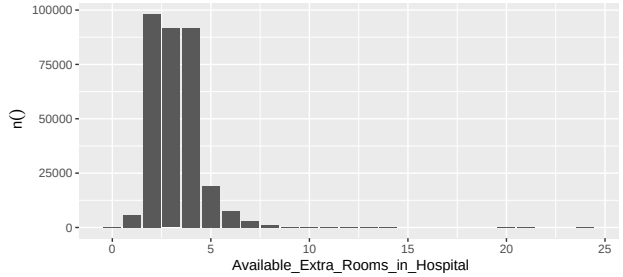
2.3.2 Visualisation of data distributions

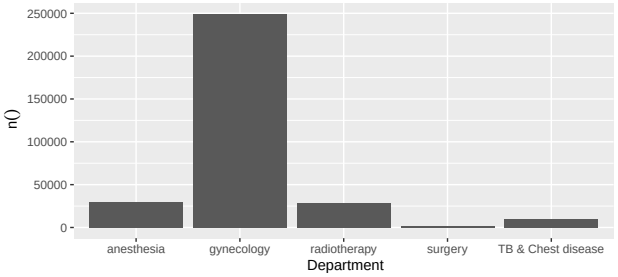
```
# Visualisation of data distributions of all features used in modelling - use dbplot as
# it works with big data in sparklyr
data %>%
    dbplot_bar(Hospital_type)
data %>%
    dbplot_bar(Hospital_region)
data %>%
    dbplot_bar(Available_Extra_Rooms_in_Hospital)
data %>%
    dbplot_bar(Department)
data %>%
    dbplot_bar(Ward_Type)
data %>%
```

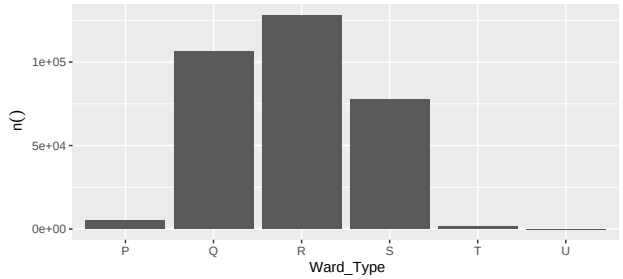
```
    dbplot_bar(Ward_Facility)
data %>%
    dbplot_bar(Bed_Grade)
data %>%
    dbplot_bar(Type_of_Admission)
data %>%
    dbplot_bar(Illness_Severity)
data %>%
    dbplot_bar(Patient_Visitors)
data %>%
    dbplot_bar(Age)
data %>%
    dbplot_bar(Admission_Deposit)
data %>%
    dbplot_bar(Stay_Days)
```

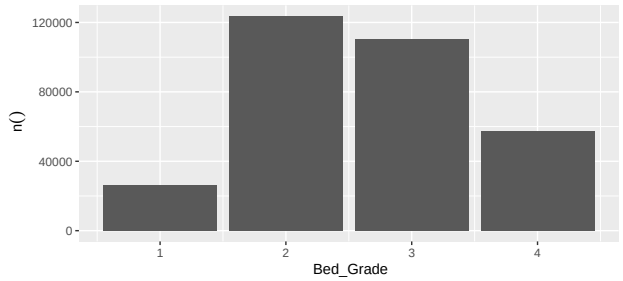


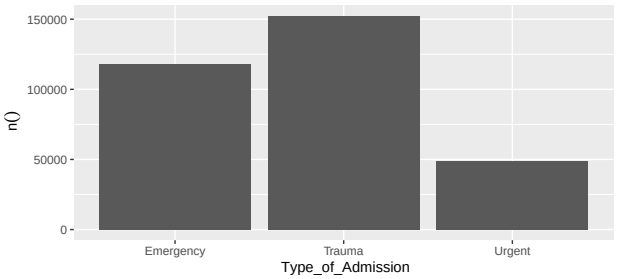


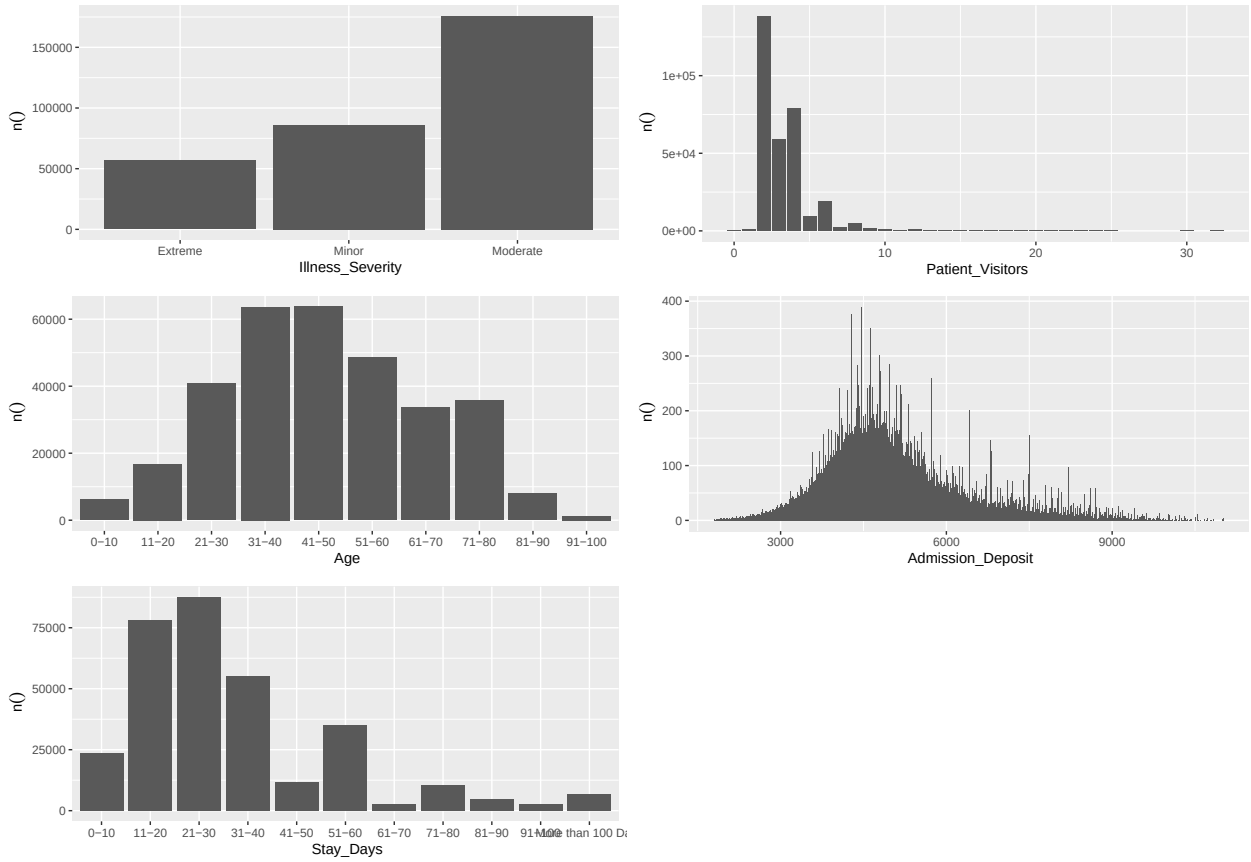












2.3.3 Description of data

There are 318438 observations, with 18 features.

Some of this information is taken from https://www.kaggle.com/datasets/arashnic/covid19-hospital-treatment. This website contains summary statistics of each feature, as well as bar chart visualisations.

Some of this information also comes from a manual inspection of the raw CSV file.

	Label	Type	Notes	Relevance/ suitability
1	case_id	Sequential, categorical, nominal	Unique ID, integers 1 to 318438	N
2	Hospital	Categorical, nominal	Integer label for each hospital, 32 categories, range: 1 to 32	N
3	Hospital_type	Categorical, nominal	Integer label for each hospital type, 7 categories, range: 0 to 6	Y

	Label	Type	Notes	Relevance/ suitability
4	Hospital_city	Categorical, nominal	Integer label for each hospital type, 13 categories, range: 1 to 13	N
5	Hospital_region	Categorical, nominal	Integer label for each hospital region, 3 categories, range: 0 to 2	Y
6	Available_Extra_Rooms_in_Hospital	Numerical, discrete (integers)	No. of extra rooms available in the Hospital, range: 0 to 24	Y
7	Department	Categorical, nominal	String with dept. overlooking the case, 5 types in data, ("TB & Chest disease", "radiotherapy", "anesthesia", "gynecology", "surgery")	Y
8	Ward_Type	Categorical, nominal	String with ward type, 6 types in data ("P", "Q", "R", "S", "T", "U")	Y
9	Ward_Facility	Categorical, nominal	String with ward facility, 6 types in data ("A", "B", "C", "D", "E", "F")	Y
10	Bed_Grade	Categorical, nominal	Integer label for condition of bed in the ward, 4 types in data, range: 1 to 4, data assumed to be NOT ordinal	Y
11	patientid	Categorical, discrete (integers)	5 or 6 digit integer label, many duplicates, range: 1 to 131624	N
12	City_Code_Patient	Categorical, nominal	Integer label for city code for the patient, 37 types in data (no '17' value), range: 1 to 38	N
13	Type_of_Admission	Categorical, nominal	String with admission type registered by the Hospital, 3 types in data ("Emergency", "Trauma", "Urgent")	Y

	Label	Type	Notes	Relevance/ suitability
14	Illness_Severity	Categorical, nominal	String with severity of the illness recorded at the time of admission, 3 types in data (“Extreme”, “Minor”, “Moderate”)	Y
15	Patient_Visitors	Numerical, discrete (integers)	Range: 0 to 32	Y
16	Age	Categorical, ordinal	String with 10-year ranges from 0 to 100, 10 types in data (“0-10”, “11-20”, “21-30”, “31-40”, “41-50”, “51-60”, “61-70”, “71-60”, “81-90”, “91-100”)	Y
17	Admission_Deposit	Numerical, discrete (integers)	Deposit at the Admission Time, range: 1800 to 11008	Y
18	Stay_Days	Categorical, ordinal	Target feature; string with stay days by the patient, 11 types in data	Y

2.3.4 Notes about data

There are no obvious errors in the data.

Most of the distributions of the features in the data are well-balanced. The ‘Available_Extra_Rooms_in_Hospital’ and ‘Patient_Visitors’ features contain small numbers of high-value data points which are probably outliers and could be removed from the data.

An assumption is made that the ‘Bed_Grade’ feature is a categorical variable as it is encoded in the data as a decimal value with one significant figure after the decimal point, but in the data the feature only takes four integer values with no obvious ordering (1.0, 2.0, 3.0, 4.0). It has 113 missing values.

An assumption is made that the ‘City_Code_Patient’ is a categorical variable although it is encoded in the data as a decimal value with one significant figure after the decimal point, but in the data the feature only takes 38 integer values, with no obvious ordering (1.0 to 38.0). It has 4352 missing values.

The ‘case_id’ and ‘patientid’ features are probably administrative ID codes so do not contain data relevant to a classification model predicting the length of stay in a hospital and are removed from the training data.

The ‘hospital’ ‘hospital_city’ and ‘City_Code_Patient’ features contain too many categories for one-hot-encoding and are not used (in this initial analysis) as they would increase the dimensionality of models significantly and would impact performance and accuracy. These features require specialised transformation techniques to allow them to be used for modelling.

The observations with missing values are completely removed from the training data, as there is not enough information to replace or impute the values with something more meaningful. The dropped observations amount to about 1.5% of the total. If this was a larger amount, then more investigation would be needed.

The 'Admission_Deposit' feature may be financial data and appears to be normally distributed with a mean around 4880.

The 'Stay_Days' feature is categorical, with 11 categories corresponding to the the length of hospital stay in numbers of days, in 10-day lengths. This requires assigning a value label (i.e. '0' to '10') for each category when fitting a model.

3 Modelling

3.1 Data preparation and transformation

```
# Split data for modelling
data_splits <- sdf_random_split(data, training = 0.8,
    testing = 0.2, seed = 42)
data_train <- data_splits$training
data_test <- data_splits$testing

# Transformation/wrangling
data_train_transform <- data_train %>%
    # Remove unused features (case_id, patientid)
select(-c(case_id, patientid)) %>%
    # Impute or remove observations with missing
    # values (Bed_Grade, City_Code_Patient)
filter_at(vars(Bed_Grade, City_Code_Patient), all_vars(!is.na(.))) %>%
    # Rescale some categorical ranges (encoded as
    # 'int' in data) to start at 0, simple -1
    # from all values, this needs to be done for
    # one-hot-encoding - Hospital, hospital_city,
    # Bed_Grade, City_Code_Patient (only one of
    # these features, Bed_Grade, is actually
    # used)
mutate(Hospital = Hospital - 1, Hospital_city = Hospital_city -
    1, Bed_Grade = Bed_Grade - 1, City_Code_Patient = City_Code_Patient -
    1)

# One-hot-encoding for selected categorical
# predictor features, do this manually
data_train_encode_hospital_type <- data_train_transform %>%
    mutate(Hospital_type_1 = ifelse(Hospital_type ==
        0, 1, 0), Hospital_type_2 = ifelse(Hospital_type ==
        1, 1, 0), Hospital_type_3 = ifelse(Hospital_type ==
        2, 1, 0), Hospital_type_4 = ifelse(Hospital_type ==
        3, 1, 0), Hospital_type_5 = ifelse(Hospital_type ==
        4, 1, 0), Hospital_type_6 = ifelse(Hospital_type ==
        5, 1, 0), Hospital_type_7 = ifelse(Hospital_type ==
        6, 1, 0))

data_train_encode_hospital_region <- data_train_encode_hospital_type %>%
    mutate(Hospital_region_1 = ifelse(Hospital_region ==
        0, 1, 0), Hospital_region_2 = ifelse(Hospital_region ==
        1, 1, 0), Hospital_region_3 = ifelse(Hospital_region ==
        2, 1, 0))
```

```
data_train_encode_department <- data_train_encode_hospital_region %>%
    mutate(Department_1 = ifelse(Department == "anesthesia",
        1, 0), Department_2 = ifelse(Department ==
        "gynecology", 1, 0), Department_3 = ifelse(Department ==
        "radiotherapy", 1, 0), Department_4 = ifelse(Department ==
        "surgery", 1, 0), Department_5 = ifelse(Department ==
        "TB & Chest disease", 1, 0))

data_train_encode_ward_type <- data_train_encode_department %>%
    mutate(Ward_Type_1 = ifelse(Ward_Type == "P", 1,
        0), Ward_Type_2 = ifelse(Ward_Type == "Q",
        1, 0), Ward_Type_3 = ifelse(Ward_Type == "R",
        1, 0), Ward_Type_4 = ifelse(Ward_Type == "S",
        1, 0), Ward_Type_5 = ifelse(Ward_Type == "T",
        1, 0), Ward_Type_6 = ifelse(Ward_Type == "U",
        1, 0))

data_train_encode_ward_facility <- data_train_encode_ward_type %>%
    mutate(Ward_Facility_1 = ifelse(Ward_Facility ==
        "A", 1, 0), Ward_Facility_2 = ifelse(Ward_Facility ==
        "B", 1, 0), Ward_Facility_3 = ifelse(Ward_Facility ==
        "C", 1, 0), Ward_Facility_4 = ifelse(Ward_Facility ==
        "D", 1, 0), Ward_Facility_5 = ifelse(Ward_Facility ==
        "E", 1, 0), Ward_Facility_6 = ifelse(Ward_Facility ==
        "F", 1, 0))

data_train_encode_bed_grade <- data_train_encode_ward_facility %>%
    mutate(Bed_Grade_1 = ifelse(Bed_Grade == 0, 1,
        0), Bed_Grade_2 = ifelse(Bed_Grade == 1, 1,
        0), Bed_Grade_3 = ifelse(Bed_Grade == 2, 1,
        0), Bed_Grade_4 = ifelse(Bed_Grade == 3, 1,
        0))

data_train_encode_admission_type <- data_train_encode_bed_grade %>%
    mutate(Type_of_Admission_1 = ifelse(Type_of_Admission ==
        "Emergency", 1, 0), Type_of_Admission_2 = ifelse(Type_of_Admission ==
        "Trauma", 1, 0), Type_of_Admission_3 = ifelse(Type_of_Admission ==
        "Urgent", 1, 0))

data_train_encode_illness_severity <- data_train_encode_admission_type %>%
    mutate(Illness_Severity_1 = ifelse(Illness_Severity ==
        "Extreme", 1, 0), Illness_Severity_2 = ifelse(Illness_Severity ==
        "Minor", 1, 0), Illness_Severity_3 = ifelse(Illness_Severity ==
        "Moderate", 1, 0))

data_train_encode_age <- data_train_encode_illness_severity %>%
    mutate(Age_1 = ifelse(Age == "0-10", 1, 0), Age_2 = ifelse(Age ==
        "11-20", 1, 0), Age_3 = ifelse(Age == "21-30",
        1, 0), Age_4 = ifelse(Age == "31-40", 1, 0),
        Age_5 = ifelse(Age == "41-50", 1, 0), Age_6 = ifelse(Age ==
            "51-60", 1, 0), Age_7 = ifelse(Age == "61-70",
            1, 0), Age_8 = ifelse(Age == "71-80", 1,
            0), Age_9 = ifelse(Age == "81-90", 1, 0),
```

```
        Age_10 = ifelse(Age == "91-100", 1, 0))
```

```
# Normalisation/scaling for all 3 numerical
# predictor features -
# Available_Extra_Rooms_in_Hospital,
# Patient_Visitors, Admission_Deposit
scale_values_extra <- data_train_encode_age %>%
    summarise(mean_extra = mean(Available_Extra_Rooms_in_Hospital),
        sd_extra = sd(Available_Extra_Rooms_in_Hospital)) %>%
    collect()

scale_values_visitors <- data_train_encode_age %>%
    summarise(mean_visitors = mean(Patient_Visitors),
        sd_visitors = sd(Patient_Visitors)) %>%
    collect()

scale_values_deposit <- data_train_encode_age %>%
    summarise(mean_deposit = mean(Admission_Deposit),
        sd_deposit = sd(Admission_Deposit)) %>%
    collect()

data_train_extra_scaled <- data_train_encode_age %>%
    mutate(Available_Extra_Rooms_in_Hospital_scaled = ((Available_Extra_Rooms_in_Hospital -
        !!scale_values_extra$mean_extra)/!!scale_values_extra$sd_extra))

data_train_visitors_scaled <- data_train_extra_scaled %>%
    mutate(Patient_Visitors_scaled = ((Patient_Visitors -
        !!scale_values_visitors$mean_visitors)/!!scale_values_visitors$sd_visitors))

data_train_deposit_scaled <- data_train_visitors_scaled %>%
    mutate(Admission_Deposit_scaled = ((Admission_Deposit -
        !!scale_values_deposit$mean_deposit)/!!scale_values_deposit$sd_deposit))

# Target feature 'Stay_Days' converted to 11 int
# value labels, 0 to 10
data_train_convert_target <- data_train_deposit_scaled %>%
    mutate(Stay_Days_converted = case_when((Stay_Days ==
        "0-10") ~ 0, (Stay_Days == "11-20") ~ 1, (Stay_Days ==
        "21-30") ~ 2, (Stay_Days == "31-40") ~ 3, (Stay_Days ==
        "41-50") ~ 4, (Stay_Days == "51-60") ~ 5, (Stay_Days ==
        "61-70") ~ 6, (Stay_Days == "71-80") ~ 7, (Stay_Days ==
        "81-90") ~ 8, (Stay_Days == "91-100") ~ 9,
        (Stay_Days == "More than 100 Days") ~ 10))
```

3.2 Details of transformed data

```
# Show transformed data ready for modelling
glimpse(data_train_convert_target)
```

```
Rows: ??
Columns: 67
```

```
Database: spark_connection
$ Hospital                                <dbl> 7, 25, 22, 31, 22, 0, 9, 21, ~
$ Hospital_type                           <int> 2, 1, 0, 5, 0, 3, 4, 6, 1, 2,~
$ Hospital_city                           <dbl> 2, 1, 5, 8, 5, 9, 0, 8, 1, 2,~
$ Hospital_region                         <int> 2, 1, 0, 1, 0, 1, 0, 1, 1, 2,~
$ Available_Extra_Rooms_in_Hospital       <int> 3, 2, 2, 1, 4, 2, 2, 2, 4, 2,~
$ Department                              <chr> "radiotherapy", "radiotherapy~
$ Ward_Type                               <chr> "R", "S", "S", "S", "Q", "R",~
$ Ward_Facility                           <chr> "F", "D", "F", "B", "F", "B",~
$ Bed_Grade                               <dbl> 1, 1, 1, 2, 2, 3, 2, 1, 0, 2,~
$ City_Code_Patient                       <dbl> 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,~
$ Type_of_Admission                       <chr> "Emergency", "Trauma", "Traum~
$ Illness_Severity                        <chr> "Extreme", "Extreme", "Extrem~
$ Patient_Visitors                        <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
$ Age                                     <chr> "51-60", "51-60", "51-60", "5~
$ Admission_Deposit                       <dbl> 4911, 5558, 4449, 6167, 5571,~
$ Stay_Days                               <chr> "0-10", "41-50", "11-20", "0-~
$ Hospital_type_1                         <dbl> 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,~
$ Hospital_type_2                         <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,~
$ Hospital_type_3                         <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
$ Hospital_type_4                         <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,~
$ Hospital_type_5                         <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,~
$ Hospital_type_6                         <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,~
$ Hospital_type_7                         <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,~
$ Hospital_region_1                       <dbl> 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,~
$ Hospital_region_2                       <dbl> 0, 1, 0, 1, 0, 1, 0, 1, 1, 0,~
$ Hospital_region_3                       <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
$ Department_1                            <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,~
$ Department_2                            <dbl> 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,~
$ Department_3                            <dbl> 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,~
$ Department_4                            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Department_5                            <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Ward_Type_1                             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Ward_Type_2                             <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,~
$ Ward_Type_3                             <dbl> 1, 0, 0, 0, 0, 1, 0, 0, 1, 1,~
$ Ward_Type_4                             <dbl> 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,~
$ Ward_Type_5                             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Ward_Type_6                             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Ward_Facility_1                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,~
$ Ward_Facility_2                         <dbl> 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,~
$ Ward_Facility_3                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Ward_Facility_4                         <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,~
$ Ward_Facility_5                         <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,~
$ Ward_Facility_6                         <dbl> 1, 0, 1, 0, 1, 0, 0, 0, 0, 0,~
$ Bed_Grade_1                             <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,~
$ Bed_Grade_2                             <dbl> 1, 1, 1, 0, 0, 0, 0, 1, 0, 0,~
$ Bed_Grade_3                             <dbl> 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,~
$ Bed_Grade_4                             <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,~
$ Type_of_Admission_1                     <dbl> 1, 0, 0, 1, 0, 0, 0, 0, 0, 1,~
$ Type_of_Admission_2                     <dbl> 0, 1, 1, 0, 1, 1, 1, 0, 0, 0,~
$ Type_of_Admission_3                     <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,~
$ Illness_Severity_1                      <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ Illness_Severity_2                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Illness_Severity_3                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

```
$ Age_1                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_2                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_3                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_4                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_5                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_6                                    <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
$ Age_7                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_8                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_9                                    <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Age_10                                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ Available_Extra_Rooms_in_Hospital_scaled <dbl> -0.1685508, -1.0241365, -1.02~
$ Patient_Visitors_scaled                  <dbl> -0.7269721, -0.7269721, -0.72~
$ Admission_Deposit_scaled                 <dbl> 0.02784173, 0.62351630, -0.39~
$ Stay_Days_converted                      <dbl> 0, 4, 1, 0, 4, 5, 3, 2, 1, 0,~
```

```
# Show correlations between four numerical/continuous features in the data (three
# scaled/normalised predictor features and the converted target feature)
ml_corr(data_train_convert_target, columns = c("Available_Extra_Rooms_in_Hospital_scaled",
    "Patient_Visitors_scaled", "Admission_Deposit_scaled", "Stay_Days_converted"))
```

```
# A tibble: 4 x 4
  Available_Extra_Rooms_in_Hospital_scaled Patient_Visitors_sc~1 Admis~2 Stay_~3
                                    <dbl>                 <dbl>   <dbl>   <dbl>
1                                   1                    0.0952 -0.144  -0.122
2                                   0.0952               1      -0.150   0.536
3                                  -0.144               -0.150   1      -0.0520
4                                  -0.122                0.536  -0.0520  1
# ... with abbreviated variable names 1: Patient_Visitors_scaled,
#   2: Admission_Deposit_scaled, 3: Stay_Days_converted
```

3.3 Fit models

3.3.1 Logistic regression classifier

```
# Logistic regression classifier

# The intercept constant is dropped to cater for
# the dummy variable trap - selected subset of
# predictor features used, numerical and manually
# one-hot encoded features
data_train_lr_model <- data_train_convert_target %>%
    ml_logistic_regression(Stay_Days_converted ~ Hospital_type_1 +
        Hospital_type_2 + Hospital_type_3 + Hospital_type_4 +
        Hospital_type_5 + Hospital_type_6 + Hospital_type_7 +
        Hospital_region_1 + Hospital_region_2 + Hospital_region_3 +
        Department_1 + Department_2 + Department_3 +
        Department_4 + Department_5 + Ward_Type_1 +
        Ward_Type_2 + Ward_Type_3 + Ward_Type_4 + Ward_Type_5 +
        Ward_Type_6 + Ward_Facility_1 + Ward_Facility_2 +
        Ward_Facility_3 + Ward_Facility_4 + Ward_Facility_5 +
        Ward_Facility_6 + Bed_Grade_1 + Bed_Grade_2 +
```

```
        Bed_Grade_3 + Bed_Grade_4 + Type_of_Admission_1 +
        Type_of_Admission_2 + Type_of_Admission_3 +
        Illness_Severity_1 + Illness_Severity_2 + Illness_Severity_3 +
        Age_1 + Age_2 + Age_3 + Age_4 + Age_5 + Age_6 +
        Age_7 + Age_8 + Age_9 + Age_10 + Available_Extra_Rooms_in_Hospital_scaled +
        Patient_Visitors_scaled + Admission_Deposit_scaled,
        family = "multinomial", fit_intercept = FALSE)

# Evaluate (metrics)
model_metrics <- ml_evaluate(data_train_lr_model, data_train_convert_target)
model_metrics$accuracy()
```

```
[1] 0.3941722
```

3.3.2 ANN multilayer perceptron classifier

```
# ANN multilayer perceptron classifier

# Selected subset of predictor features in first
# layer, numerical and manually one-hot encoded
# features - the last feature in each
# one-hot-encoded set is removed to get around
# the dummy variable trap - the 'Hospital_type'
# feature is not used as it causes a Java stack
# overflow error

data_train_ann_model <- data_train_convert_target %>%
    ml_multilayer_perceptron_classifier(Stay_Days_converted ~
        Hospital_region_1 + Hospital_region_2 + Department_1 +
            Department_2 + Department_3 + Department_4 +
            Ward_Type_1 + Ward_Type_2 + Ward_Type_3 +
            Ward_Type_4 + Ward_Type_5 + Ward_Facility_1 +
            Ward_Facility_2 + Ward_Facility_3 + Ward_Facility_4 +
            Ward_Facility_5 + Bed_Grade_1 + Bed_Grade_2 +
            Bed_Grade_3 + Type_of_Admission_1 + Type_of_Admission_2 +
            Illness_Severity_1 + Illness_Severity_2 +
            Age_1 + Age_2 + Age_3 + Age_4 + Age_5 +
            Age_6 + Age_7 + Age_8 + Age_9 + Available_Extra_Rooms_in_Hospital_scaled +
            Patient_Visitors_scaled + Admission_Deposit_scaled,
        layers = c(35, 8, 8, 11))

# Evaluate (metrics)
model_metrics <- ml_evaluate(data_train_ann_model,
    data_train_convert_target)
model_metrics
```

```
# A tibble: 1 x 1
  Accuracy
     <dbl>
1    0.368
```

3.3.3 Random forest classifier

```
# Random forest classifier

# Selected subset of predictor features used,
# numerical and manually one-hot encoded features
# - the last feature in each one-hot-encoded set
# is removed to get around the dummy variable
# trap

data_train_rf_model <- data_train_convert_target %>%
    ml_random_forest_classifier(Stay_Days_converted ~
        Hospital_type_1 + Hospital_type_2 + Hospital_type_3 +
            Hospital_type_4 + Hospital_type_5 + Hospital_type_6 +
            Hospital_region_1 + Hospital_region_2 +
            Department_1 + Department_2 + Department_3 +
            Department_4 + Ward_Type_1 + Ward_Type_2 +
            Ward_Type_3 + Ward_Type_4 + Ward_Type_5 +
            Ward_Facility_1 + Ward_Facility_2 + Ward_Facility_3 +
            Ward_Facility_4 + Ward_Facility_5 + Bed_Grade_1 +
            Bed_Grade_2 + Bed_Grade_3 + Type_of_Admission_1 +
            Type_of_Admission_2 + Illness_Severity_1 +
            Illness_Severity_2 + Age_1 + Age_2 + Age_3 +
            Age_4 + Age_5 + Age_6 + Age_7 + Age_8 +
            Age_9 + Available_Extra_Rooms_in_Hospital_scaled +
            Patient_Visitors_scaled + Admission_Deposit_scaled,
        num_trees = 20)

# Evaluate (metrics)
model_metrics <- ml_evaluate(data_train_rf_model, data_train_convert_target)
model_metrics
```

```
# A tibble: 1 x 1
  Accuracy
     <dbl>
1    0.320
```

3.4 Disconnection

```
# Disconnect from data source
spark_disconnect(sc)
```

4 Results, conclusions and further work

The three models used in this analysis have been chosen with explainability as a main criteria, and all are standard well-known machine learning models that support multinomial classification, required for this analysis. The three models work in quite different ways to allow scope for exploring and comparing different approaches. The gradient-boosted trees classifier model was considered, but multinomial classification is not supported in this model in the Sparklyr library.

To compare the performance of all three models, the Accuracy metric is used as this is appropriate for a multinomial classification approach, and is supported by all the models.

Results:

Model	Accuracy
Logistic regression classifier	0.394
ANN multilayer perceptron classifier	0.368
Random forest classifier	0.320

The accuracy of all the selected models is poor, although the logistic regression classifier performs best.

Several areas of further work could be investigated in an attempt to improve this:

- Carry out hyperparameter tuning on the models with crossfold validation on subsets of the training data, as this analysis only uses default values for hyperparameters
- Use more predictor features - this analysis has not used features with a large number of categories (hospital' 'hospital_city' and 'City_Code_Patient') as this generates very large one-hot-encoding sets, but methods could be investigated to allow the inclusion of these features (and also new features not included in the original data)
- Explore other types of model such as support vector machine or naive bayes
- Carry out feature engineering with imbalanced or skewed data distributions, co-linearities/correlations (although this would require dealing with the categorical features in a special way, such as using Cramér's V), information gain and principal component analysis (dimensionality reduction) of predictor features

If these areas of work resulted in any substantial improvements in the models, the best would be chosen and the performance of the model evaluated with the test data.

Appendix

Presentation URL

https://media.ed.ac.uk/media/Kaltura+Capture+recording+-+December+9th+2022%2C+10A37A00+am/1_anxv4l37