PATIENT SURVIVABILITY

Submitted by

ADITYA MISHRA [RA2011047010016]
ABHAY DIXIT [RA2011047010023]
NITIN KUMAR [RA2011047010029]
UMANG KUMAR [RA2011047010047]
ANIMESH AGNIHOTRI [RA2011047010069]
RIBHU BANERJEE [RA2011047010056]

Under the guidance of
Dr. R. Babu
(Assistant Professor, Department of Computational Intelligence)

in partial fulfillment for the award of the degree
Of
BACHELOR OF TECHNOLOGY
in
COMPUTATIONAL INTELLIGENCE
of
FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District
JUNE 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"PAITENT SURVIVABILITY"** is the bonafide work of **"ADITYA MISHRA [RA2011047010016], ABHAY DIXIT [RA2011047010023], NITIN KUMAR [RA2011047010029], UMANG KUMAR [RA2011047010047], ANIMESH AGNIHOTRI [RA2011047010069], RIBHU BANERJEE [RA2011047010056]",** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.R.BABU
GUIDE
Assistant Professor
Dept. of Computational Intelligence

SIGNATURE

Dr. ANNIE UTHRA
HEAD OF THE DEPARTMENT
Dept. of Computational Intelligence

Signature of the Internal Examiner

Signature of the External Exam

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, **Dr. R. Babu**, his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing the project. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this project work.

ADITYA MISHRA
ABHAY DIXIT
NITIN KUMAR
UMANG KUMAR
ANIMESH AGNIHOTRI
RIBHU BANERJEE

TABLE OF CONTENTS

Chapter 1: INTRODUCTION

1.1 Objective:

A hospital has been trying to improve its care conditions by looking at historic survival of the patients. They tried looking at their data but could not identify the main factors leading to high survivals. This model helps in predicting the chances of survival of a patient after 1 year of treatment by using various Machine Learning Algorithms like Logistic Regression, Random Forest and BorutaPy.

1.2 Problem Statement:

Admission to an intensive care unit (ICU) is lifesaving for some patients, but for many, the admission carries high expectations and financial costs and fails to provide desirable outcomes. Patients who receive intensive care have a mortality rate of about 20%, and the costs of this care comprise about 4% of the U.S. health care budget. This model helps in predicting the chances of survival of a patient after 1 year of treatment.

1.3 Proposed Solution:

The proposed solution is to build a Machine Learning Model which will predict the survival percentage of patients after a year of treatment using Logistic Regression, Random Forest and BorutaPy in Python.

Chapter 2

Data Description:

- ID_Patient_Care_Situation: Care situation of a patient during treatment
- Diagnosed_Condition: The diagnosed condition of the patient
- ID_Patient: Patient identifier number
- Treatment_with_drugs: Class of drugs used during treatment
- Survived_1_year: If the patient survived after one year (0 means did not survive; 1 means survived)
- Patient_Age: Age of the patient
- Patient_Body_Mass_Index: A calculated value based on the patient's weight, height, etc.
- Patient_Smoker: If the patient was a smoker or not
- Patient_Rural_Urban: If the patient stayed in Rural or Urban part of the country
- Previous_Condition: Condition of the patient before the start of the treatment (This variable is splitted into 8 columns - A, B, C, D, E, F, Z and Number_of_prev_cond. A, B, C, D, E, F and Z are the previous conditions of the patient.

Algorithm:

1. Load data
2. Data Preparation
3. Understanding the data: EDA
4. Pre-processing the data
5. Prepare train and test datasets
6. Choose a model
7. Train the model
8. Evaluate the model (F1-score calculation)
9. Optimize: repeat steps 5 – 7 using different algorithms

Chapter 3: TOOLS AND SOFTWARES USED

Tools Description:

Google Colab

● Colab is basically a free Jupyter notebook environment running wholly in the cloud.

● Most importantly, Colab does not require a setup, plus the notebooks that you will create can be simultaneously edited by your team members – in a similar manner to how you edit documents in Google Docs.

● The greatest advantage is that Colab supports the most popular machine learning libraries which can be easily loaded in your notebook. As a developer, you can perform the following using Google Colab.

● Write and execute code in Python

● Create/Upload/Share notebooks

● Import/Save notebooks from/to Google Drive

● Import/Publish notebooks from GitHub

● Import external datasets

● Integrate PyTorch, TensorFlow, Keras, OpenCV

● Free Cloud service with free GPU

Python:

Python is a high-level, general-purpose and a very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry. Python Programming Language is very well suited for Beginners, also for experienced programmers with other programming languages like C++ and Java.

Below are some facts about Python Programming Language:

1. Python is currently the most widely used multi-purpose, high-level programming language.
2. Python allows programming in Object-Oriented and Procedural paradigms.
3. Python programs generally are smaller than other programming languages like Java.

Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.

1. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.
2. The biggest strength of Python is huge collection of standard library which can be used for the following.
    - Machine Learning
    - GUI Applications (like Kivy, Tkinter, PyQt etc.)
    - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
    - Image processing (like OpenCV, Pillow)
    - Web scraping (like Scrapy, BeautifulSoup, Selenium)
    - Test frameworks
    - Multimedia
    - Scientific computing
    - Text processing and many more.

Chapter 4: RESULTS AND DISCUSSION

Code Implementation:

```
[1] import pandas as pd
    pharma_data=pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/Training_
```

```
[2] import pandas as pd   # package for data analysis
    import numpy as np    # package for numerical computations
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split   # To split the dataset into train and test set
    from sklearn.linear_model import LogisticRegression     # Logistic regression model
    from sklearn.metrics import f1_score    # for model evaluation
```

```
[3] data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/Training_set_
```

```
[4] data.head()
```

	ID_Patient_Care_Situation	Diagnosed_Condition	Patient_ID	Treated_with_drugs	Patient_Age	Patient_Body_M
0	22374	8	3333	DX6	56	
1	18164	5	5740	DX2	36	
2	6283	23	10446	DX6	48	
3	5339	51	12011	DX1	5	

```
[1] import pandas as pd
    pharma_data=pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/Training_set_begs.csv')
```

```
[2] import pandas as pd   # package for data analysis
    import numpy as np    # package for numerical computations
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split   # To split the dataset into train and test set
    from sklearn.linear_model import LogisticRegression     # Logistic regression model
    from sklearn.metrics import f1_score    # for model evaluation
```

```
[3] data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/Training_set_begs.csv')
```

```
[4] data.head()
```

	ID_Patient_Care_Situation	Diagnosed_Condition	Patient_ID	Treated_with_drugs	Patient_Age	Patient_Body_Mass_Index	Patient_S
0	22374	8	3333	DX6	56	18.479385	
1	18164	5	5740	DX2	36	22.945566	
2	6283	23	10446	DX6	48	27.510027	
3	5339	51	12011	DX1	5	19.139976	

```
[1] import pandas as pd
    pharma_data=pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/T
```

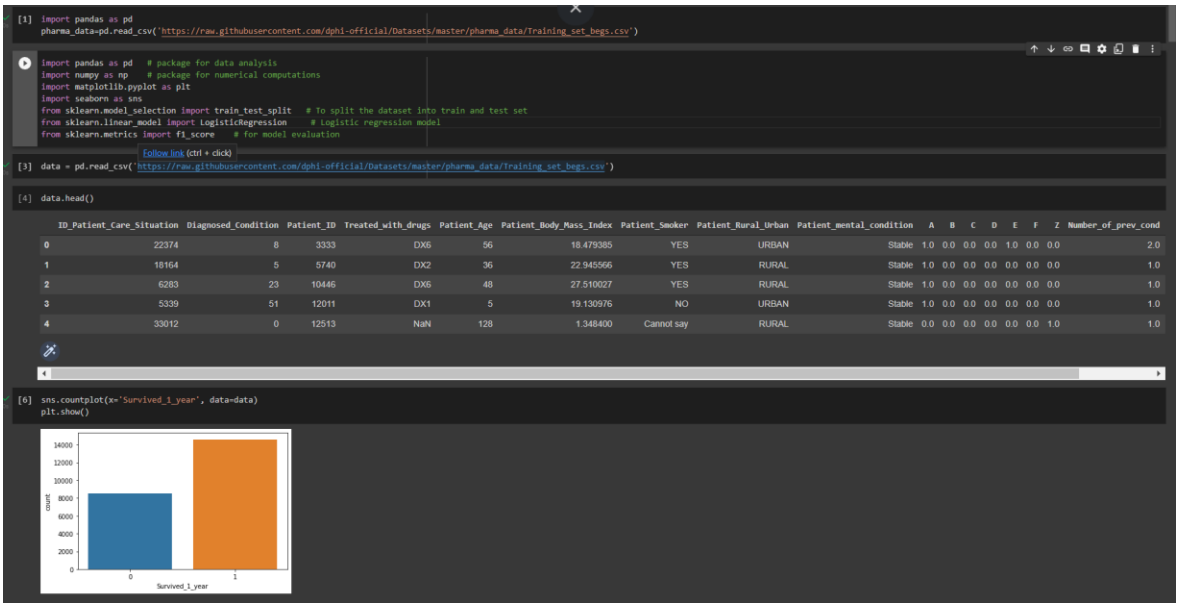
```
import pandas as pd   # package for data analysis
import numpy as np    # package for numerical computations
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split   # To split the dataset into train and test set
from sklearn.linear_model import LogisticRegression     # Logistic regression model
from sklearn.metrics import f1_score    # for model evaluation
```

Follow link (ctrl + click)

```
data = pd.read_csv('https://raw.githubusercontent.com/dphi-official/Datasets/master/pharma_data/Traini
```

```
[4] data.head()
```

	ID_Patient_Care_Situation	Diagnosed_Condition	Patient_ID	Treated_with_drugs	Patient_Age	Patien
0	22374	8	3333	DX6	56	
1	18164	5	5740	DX2	36	
2	6283	23	10446	DX6	48	
3	5339	51	12011	DX1	5	
4	33012	0	12513	NaN	128	



```
[7] # getting only the numerical features
    numeric_features = data.select_dtypes(include=[np.number])    # select_dtypes helps you to select data of p    r types
    numeric_features.columns
```

```
Index(['ID_Patient_Care_Situation', 'Diagnosed_Condition', 'Patient_ID',
       'Patient_Age', 'Patient_Body_Mass_Index', 'A', 'B', 'C', 'D', 'E', 'F',
       'Z', 'Number_of_prev_cond', 'Survived_1_year'],
      dtype='object')
```

```
[8] numeric_data=data[['Diagnosed_Condition', 'Patient_Age', 'Patient_Body_Mass_Index', 'Number_of_prev_cond', 'Survived_1_year']]  #keeping in the target varibale for analysis purposes
    numeric_data.head()
```

	Diagnosed_Condition	Patient_Age	Patient_Body_Mass_Index	Number_of_prev_cond	Survived_1_year
0	8	56	18.479385	2.0	0
1	5	36	22.945566	1.0	1
2	23	48	27.510027	1.0	0
3	51	5	19.130976	1.0	1
4	0	128	1.348400	1.0	1

```
[10] # Checking the null values in numerical columns
     numeric_data.isnull().sum()
```

```
Diagnosed_Condition          0
Patient_Age                  0
Patient_Body_Mass_Index      0
Number_of_prev_cond       1235
Survived_1_year              0
dtype: int64
```

```
[11] data['Number_of_prev_cond'] = data['Number_of_prev_cond'].fillna(data['Number_of_prev_cond'].mode()[0])  # filling the missing value of 'Number_of_prev_cond'
     numeric_data['Number_of_prev_cond']=data['Number_of_prev_cond']
     numeric_data.isnull().sum()
```

```
Diagnosed_Condition       0
Patient_Age               0
Patient_Body_Mass_Index   0
Number_of_prev_cond       0
Survived_1_year           0
dtype: int64
```

```
[12] numeric_data.describe()
```

	Diagnosed_Condition	Patient_Age	Patient_Body_Mass_Index	Number_of_prev_cond	Survived_1_year
count	23097.000000	23097.000000	23097.000000	23097.000000	23097.000000
mean	26.413127	33.209768	23.454820	1.710352	0.632247
std	15.030865	19.549882	3.807661	0.768216	0.482204
min	0.000000	0.000000	1.089300	1.000000	0.000000
25%	13.000000	16.000000	20.205550	1.000000	0.000000
50%	26.000000	33.000000	23.386199	2.000000	1.000000
75%	39.000000	50.000000	26.788154	2.000000	1.000000
max	52.000000	149.000000	29.999579	5.000000	1.000000

```
numeric_data=numeric_data.drop(['Survived_1_year'], axis=1)
colormap = sns.diverging_palette(10, 220, as_cmap = True)
sns.heatmap(numeric_data.corr(),
            cmap = colormap,
            square = True,
            annot = True)
plt.show()
```

```
[14] data.isnull().sum()
```

```
ID_Patient_Care_Situation       0
Diagnosed_Condition             0
Patient_ID                      0
Treated_with_drugs             13
Patient_Age                     0
Patient_Body_Mass_Index         0
Patient_Smoker                  0
Patient_Rural_Urban             0
Patient_mental_condition        0
A                            1235
B                            1235
C                            1235
D                            1235
E                            1235
F                            1235
Z                            1235
Number_of_prev_cond             0
Survived_1_year                 0
dtype: int64
```

```
[15] data['Treated_with_drugs']=data['Treated_with_drugs'].fillna(data['Treated_with_drugs'].mode()[0])
```

```
categorical_data = data.drop(numeric_data.columns, axis=1)    # dropping the numerical columns from the dataframe 'data'
categorical_data.drop(['Patient_ID', 'ID_Patient_Care_Situation'], axis=1, inplace = True)    # dropping the id columns form the dataframe 'categorical data'
categorical_data.head()    # Now we are left with categorical columns only. take a look at first five observaiton
```

	Treated_with_drugs	Patient_Smoker	Patient_Rural_Urban	Patient_mental_condition	A	B	C	D	E	F	Z	Survived_1_year
0	DX6	YES	URBAN	Stable	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0
1	DX2	YES	RURAL	Stable	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	DX6	YES	RURAL	Stable	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	DX1	NO	URBAN	Stable	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	DX6	Cannot say	RURAL	Stable	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1

```
[18] categorical_data.nunique()   # nunique() return you the number of unique values in each column/feature
```

```
Treated_with_drugs          32
Patient_Smoker               3
Patient_Rural_Urban          2
Patient_mental_condition     1
A                            2
B                            2
C                            2
D                            2
E                            2
F                            2
Z                            2
Survived_1_year              2
dtype: int64
```

```
[20] drugs = data['Treated_with_drugs'].str.get_dummies(sep=' ') # split all the entries separated by space and create dummy variable
drugs.head()
```

	DX1	DX2	DX3	DX4	DX5	DX6
0	0	0	0	0	0	1
1	0	1	0	0	0	0
2	0	0	0	0	0	1
3	1	0	0	0	0	0
4	0	0	0	0	0	1

```
[21] data = pd.concat([data, drugs], axis=1)     # concat the two dataframes 'drugs' and 'data'
data = data.drop('Treated_with_drugs', axis=1)    # dropping the column 'Treated_with_drugs' as its values are now splitted into different columns

data.head()
```

	ID_Patient_Care_Situation	Diagnosed_Condition	Patient_ID	Patient_Age	Patient_Body_Mass_Index	Patient_Smoker	Patient_Rural_Urban	Patient_mental_condition	A	B	...	F	Z	Number_of_prev_cond	Survived_1_year	DX1	DX2	D
0	22374	8	3333	56	18.479385	YES	URBAN	Stable	1.0	0.0	...	0.0	0.0	2.0	0	0	0	
1	18164	5	5740	36	22.945566	YES	RURAL	Stable	1.0	0.0	...	0.0	0.0	1.0	1	0	1	
2	6283	23	10446	48	27.510027	YES	RURAL	Stable	1.0	0.0	...	0.0	0.0	1.0	0	0	0	
3	5339	51	12011	5	19.130976	NO	URBAN	Stable	1.0	0.0	...	0.0	0.0	1.0	1	1	0	
4	33012	0	12513	128	1.348400	Cannot say	RURAL	Stable	0.0	0.0	...	0.0	1.0	1.0	1	0	0	

5 rows × 23 columns

```
[22] data.Patient_Smoker.value_counts()
```

```
NO            13246
YES            9838
Cannot say       13
Name: Patient_Smoker, dtype: int64
```

```
[23] data.Patient_Smoker[data['Patient_Smoker'] == "Cannot say"] = 'NO'    # we already know 'NO' is the mode so directly changing the values 'Cannot say' to 'NO'
```

```
[24] data.drop('Patient_mental_condition', axis = 1, inplace=True)
```

```
[25] data = pd.get_dummies(data, columns=['Patient_Smoker', 'Patient_Rural_Urban'])
```

```
[33] X = data.drop('Survived_1_year',axis = 1)
     y = data['Survived_1_year']

[34] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

[35] model = LogisticRegression(max_iter = 1000)     # The maximum number of iterations will be 1000. This will help you prevent from convergence warning.
     model.fit(X_train,y_train)

     LogisticRegression(max_iter=1000)
```

Logistic Regression Accuracy

```
[36] pred = model.predict(X_test)

     print(f1_score(y_test,pred))

     0.787275064267352 3
```

Random Forest Accuracy

```
[38] from sklearn.feature_selection import SelectFromModel
     from sklearn.metrics import accuracy_score, f1_score
     from sklearn.ensemble import RandomForestClassifier

[39] forest = RandomForestClassifier(random_state=1, n_estimators=1000, max_depth=5)

     forest.fit(X_train, y_train)

     RandomForestClassifier(max_depth=5, n_estimators=1000, random_state=1)

[40] y_pred = forest.predict(X_test)

     fscore = f1_score(y_test ,y_pred)
     fscore

     0.8220447284345048

[41] !pip install Boruta
```

```
     from boruta import BorutaPy

[43] boruta_selector = BorutaPy(forest, n_estimators='auto', verbose=2, random_state=1)   # initialize the boruta selector
     boruta_selector.fit(np.array(X_train), np.array(y_train))       # fitting the boruta selector to get all relavent features. NOTE: BorutaPy accepts numpy arrays only.
```

```
[44] print("Selected Features: ", boruta_selector.support_)    # check selected features
     print("Ranking: ",boruta_selector.ranking_)               # check ranking of features
     print("No. of significant features: ", boruta_selector.n_features_)

     Selected Features:  [ True False  True  True  True  True False  True False False False  True
       True  True  True  True  True  True  True  True  True  True]
     Ranking:  [1 2 1 1 1 1 3 1 4 5 6 1 1 1 1 1 1 1 1 1 1 1]
     No. of significant features:  17

[45] selected_rfe_features = pd.DataFrame({'Feature':list(X_train.columns),
                                           'Ranking':boruta_selector.ranking_})
     selected_rfe_features.sort_values(by='Ranking')
```

BorutaPy Accuracy

```
[48] y_important_pred = rf_important.predict(X_important_test)
     rf_imp_fscore = f1_score(y_test, y_important_pred)

[49] print(rf_imp_fscore)

     0.8578215134034612

[49]

[50] from sklearn.model_selection import GridSearchCV
     # Create the parameter grid based on the results of random search
     param_grid = {
         'bootstrap': [True, False],
         'max_depth': [5, 10, 15],
         'n_estimators': [500, 1000]}

[51] rf = RandomForestClassifier(random_state = 1)

     # Grid search cv
     grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                                cv = 2, n_jobs = -1, verbose = 2)

[52] grid_search.fit(X_important_train, y_train)

     Fitting 2 folds for each of 12 candidates, totalling 24 fits
     GridSearchCV(cv=2, estimator=RandomForestClassifier(random_state=1), n_jobs=-1,
                  param_grid={'bootstrap': [True, False], 'max_depth': [5, 10, 15],
                              'n_estimators': [500, 1000]},
                  verbose=2)

[53] grid_search.best_params_

     {'bootstrap': True, 'max_depth': 15, 'n_estimators': 500}

[54] pred = grid_search.predict(X_important_test)
```

Accuracy after Optimisation

```
[55] f1_score(y_test,   pred)

     0.8657267539442766
```

RESULT:

Logistic Regression Accuracy

```
[36] pred = model.predict(X_test)
```

```
[37] print(f1_score(y_test,pred))
```

```
0.7872750642673523
```

Random Forest Accuracy

```
[40] y_pred = forest.predict(X_test)

     fscore = f1_score(y_test ,y_pred)
     fscore
```

```
0.8220447284345048
```

BorutaPy Accuracy

```
[48] y_important_pred = rf_important.predict(X_important_test)
     rf_imp_fscore = f1_score(y_test, y_important_pred)
```

```
[49] print(rf_imp_fscore)
```

```
0.8578215134034612
```

Accuracy after Optimisation

```
[55] f1_score(y_test,   pred)
```

```
0.8657267539442766
```

Chapter 5: CONCLUSION

Admission to an intensive care unit (ICU) is lifesaving for some patients, but for many, the admission carries high expectations and financial costs and fails to provide desirable outcomes. Patients who receive intensive care have a mortality rate of about 20%, and the costs of this care comprise about 4% of the U.S. health care budget.

In a study of Medicare recipients, treatment intensity and expenses increased between the mid-1980s and 1999 but without any increase in survivorship; per capita ICU expenses were higher for patients who did not survive the ICU.

Use of the ICU in patients' final stages of life has increased in proportion since then, and the demand for critical care is likely to continue as the relative proportion of elderly patients in the population rises.

The model was successfully executed and it is observed that the average rate of survivability of a patient is: 86%.