CHAPTER 1

INTRODUCTION

1.1 Introduction

Modern work environments demand proactive strategies to ensure worker safety amidst evolving risks. Traditional risk assessment methods often fall short in adapting to dynamic workplace challenges. To address this, our study proposes an innovative AI-powered approach. By integrating advanced machine learning algorithms and predictive analytics, our system analyzes diverse data points to predict individual worker risk levels. This proactive model enables preemptive interventions and targeted safety measures, revolutionizing risk management practices and fostering a safer working environment across various industries.

Through empirical validation, our research showcases the transformative potential of AI in enhancing occupational safety. By offering a data-driven and proactive approach to risk assessment, our system represents a significant advancement in safeguarding the well-being of workers. In summary, our study advocates for the adoption of AI-driven predictive risk assessment as a fundamental element of modern occupational safety initiatives, paving the way for a future where proactive risk management becomes standard practice.

1.2 Problem Definition

In recent years, the integration of Artificial Intelligence (AI) into occupational safety management has emerged as a promising avenue for enhancing workplace safety. One crucial aspect of this application is the prediction of worker risk levels, which involves leveraging advanced AI algorithms to analyze a multitude of variables and discern patterns that may contribute to potential hazards. The proactive identification of risks enables organizations to preemptively address and mitigate workplace dangers, thereby fostering a safer and healthier working environment. This innovative approach harnesses the power of machine learning and data analytics to predict worker risk levels with a higher degree

of accuracy and efficiency than traditional methods. By incorporating AI processes into risk assessment, organizations can not only prioritize safety protocols but also optimize resource allocation and improve overall occupational health and safety outcomes. This research aims to explore and develop a comprehensive AI-based framework for predicting worker risk levels, contributing to the ongoing evolution of workplace safety practices in the era of technological advancement.

CHAPTER 2

LITERATURE REVIEW

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Title: Dynamic mortality prediction using machine learning techniques for acute cardiovascular cases

Author: Oleg Metskera, Sergey Sikorsky

Year : 2018

of model components and their connection is developed. The model structure identification assimilates the patient condition, treatment phases in treatment dynamic. It provides the prediction of better model structure on following steps and request for necessary data to improve the forecast for more informed decision-making. The dynamic data extracted directly from medical information system were analysed, that is very close to the real process. Using machine learning methods it is possible to make an early prediction of mortality risks.

Title: Mortality Prediction in ICU Patients Using Machine Learning Models

Author: Akhyar Ali Khan, Rehan Liaqat

Year: 2021

Effective utilization of limited intensive care unit (ICU) allocations is a challenging task for medical experts to save precious human lives. The prolonged ICU stay of relatively secure patients and patients with low chances of recovery can cause life-threatening effects on patients waiting for ICU accommodation. Machine learningbased techniques for early prediction of mortality can help in this regard. This paper presents two mortality prediction models using the support vector machine (SVM) and linear discriminant analysis. The proposed models use clinical data of the ICU patients for early prediction of mortality. Distribution filtering is performed using the chi-square distribution during pre-processing. A subset of the publicly available Medical Information Mart for Intensive Care (MIMIC-III v1.4) dataset is used for the evaluation of the proposed models. The problem of class imbalance is handled by synthetic minority oversampling technique. The comparison of obtained results is performed with existing SVM and multiple logistic regression models to show the effectiveness of the proposed models. The results of the study can be helpful for clinical experts for better decision making regarding the utilization of ICU allocations.

Title: Length-of-stay and mortality prediction for a major hospital through interpretable machine learning

Author: Dimitris Bertsimas, Jean Pauphilet

Importance: Understanding the discharge process at a hospital level is key in improving efficiency and quality of care. Objective: Investigate how machine learning can help anticipate various aspects of patient discharges, from predicting length-of-stay to discharge destination or hospital mortality. Design: Retrospective study performed on inpatients admitted at Beth Israel Deaconess Medical Center between January 2017 and August 2018. Setting: Single-center study in a large academic medical center in the Boston area. Participants: We included inpatients admitted at BIDMC between January 2017 and August 2018, excluding patients admitted into psychiatry, obstetrics and newborns. The

final cohort consisted of 48,770 unique admissions (32,829 unique patients).

Title: Identifying Predictors of COVID-19 Mortality Using Machine Learning
Author: Tsz-Kin Wan, Rui-Xuan Huang
Year: 2022

(1) Background: Coronavirus disease 2019 (COVID-19) is a dominant, rapidly spreading respiratory disease. However, the factors influencing COVID-19 mortality still have not been confirmed. The pathogenesis of COVID-19 is unknown, and relevant mortality predictors are lacking. This study aimed to investigate COVID-19 mortality in patients with pre-existing health conditions and to examine the association between COVID-19 mortality and other morbidities. (2) Methods: De-identified data from 113,882, including 14,877 COVID-19 patients, were collected from the UK Biobank. Different types of data, such as disease history and lifestyle factors, from the COVID-19 patients, were input into the following three machine learning models: Deep Neural Networks (DNN), Random Forest Classifier (RF), eXtreme Gradient Boosting classifier (XGB) and Support Vector Machine (SVM). The Area under the Curve (AUC) was used to measure the experiment result as a performance metric. (3) Results: Data from 14,876 COVID-19 patients were input into the machine learning model for risk-level mortality prediction, with the predicted risk level ranging from 0 to 1. Of the three models used in the experiment, the RF model achieved the best result, with an AUC value of 0.86 (95% CI 0.84–0.88). (4) Conclusions: A risk-level prediction model for COVID-19 mortality was developed. Age, lifestyle, illness, income, and family disease history were identified as important predictors of COVID-19 mortality. The identified factors were related to COVID-19 mortality.

Title: Mortality Prediction in the ICU
Author: Joon Lee, Joel A. Dubin and David M. Maslove
Year: 2016

Patients admitted to the ICU suffer from critical illness or injury and are at high risk of dying. ICU mortality rates differ widely depending on the underlying disease process, with death rates as low as 1 in 20 for patients admitted following elective surgery, and as high

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Data Pre-processing

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different **data cleaning** tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, **missing values** and it able to **more quickly clean data**. It wants to **spend less time cleaning data**, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to

deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

MODULE DIAGRAM

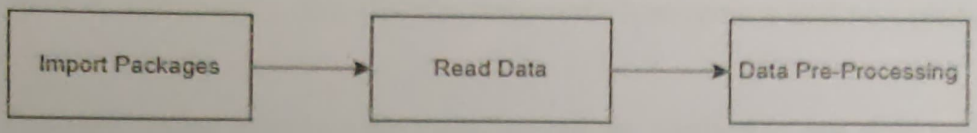

Fig 4.1 Module Diagram for Data Pre-Processing

GIVEN INPUT EXPECTED OUTPUT

input : data

output : removing noisy data

4.2 Data analysis of visualization

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

MODULE DIAGRAM

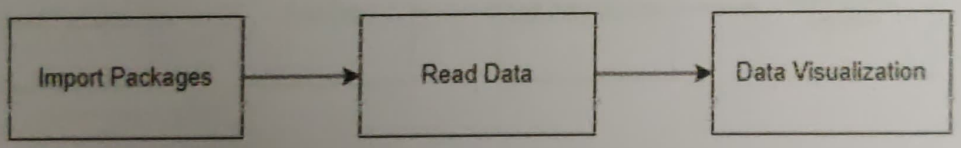

Fig 4.2 Module Diagram for Data Analysis of Visualisation

GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

4.3 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning

algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

SVMs are used in applications like handwriting recognition, intrusion detection, face detection, email classification, gene classification, and in web pages. This is one of the reasons we use SVMs in machine learning. It can handle both classification and regression on linear and non-linear data.

MODULE DIAGRAM

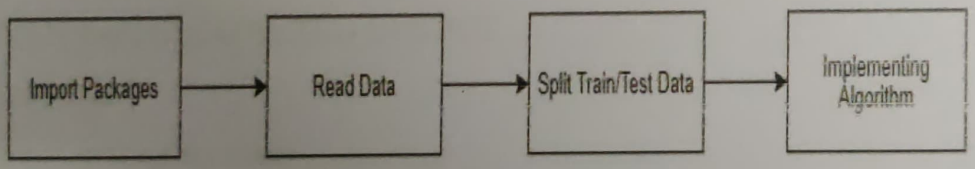

Fig 4.3 Module Diagram for Support Vector Machine

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

4.4 K-Nearest Neighbor Classifier

The KNN algorithm can compete with the most accurate models because it makes highly accurate predictions. Therefore, you can use the KNN algorithm for applications that require

high accuracy but that do not require a human-readable model. The quality of the predictions depends on the distance measure. It is a simple, supervised machine learning algorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows. Doesn't work well with a large dataset: Since KNN is a distance-based algorithm, the cost of calculating distance between a new point and each existing point is very high which in turn degrades the performance of the algorithm.

MODULE DIAGRAM

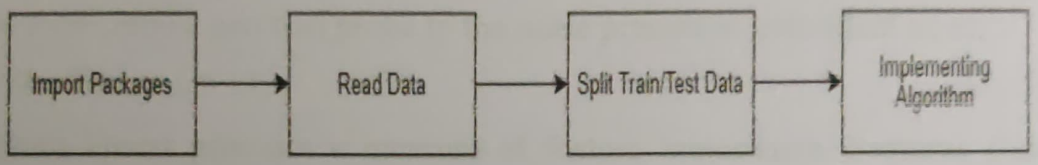

Fig 4.4 Module Diagram for KNN Classifier

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

4.5 Random Forest Classifier

Random Forest is an ensemble learning algorithm that can be used for both classification and regression tasks. It builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Here's a step-by-step explanation of how Random Forest works:

Bootstrapping:

Random Forest starts by creating multiple subsets of the original dataset through a process called bootstrapping. This involves randomly sampling the data with replacement, creating new datasets of the same size as the original.

Building Decision Trees:

For each subset, a decision tree is constructed. Decision trees are built by selecting the best feature from a random subset of features at each node, considering various criteria such as Gini impurity for classification or mean squared error for regression.

Voting (Classification) or Averaging (Regression):

Once all the trees are built, they "vote" for the class (in classification) or provide a prediction (in regression) for a new data point. For classification, the class that receives the most votes becomes the predicted class. For regression, the predictions are averaged.

Reducing Overfitting:

One of the key advantages of Random Forest is that it reduces overfitting. Each tree in the forest is trained on a different subset of the data, and by averaging or voting, the model becomes more robust and less prone to the noise present in individual trees.

Feature Importance:

Random Forest provides a measure of feature importance. Features that are more frequently used for splitting in the trees are considered more important. This information can be valuable for understanding the underlying patterns in the data.

Tuning Parameters:

Random Forest has several hyperparameters that can be tuned to optimize its performance, including the number of trees in the forest, the maximum depth of each tree, and the number of features considered at each split.

Advantages of Random Forest:

Reduces overfitting.

Handles missing values well.

Provides feature importance.

Can handle large datasets with high dimensionality.

Disadvantages:

Can be computationally expensive.

The model may become less interpretable with a large number of trees.

Random Forest is a versatile and powerful algorithm that is widely used in practice, especially in situations where interpretability is not the primary concern and high

predictive accuracy is desired.

Top of Form

MODULE DIAGRAM

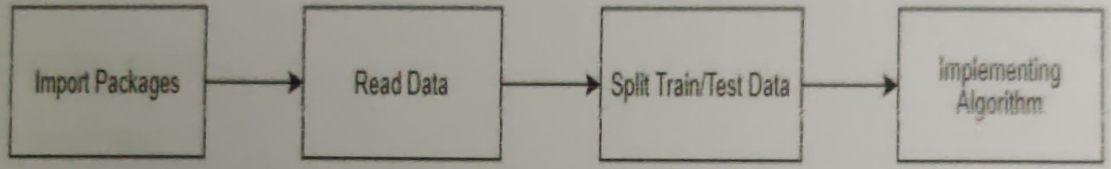

Fig 4.5 Module Diagram for KNN Classifier

GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

A.2 SOURCE CODE

Data pre-processing

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('hospital.csv')
data.head()
data.shape
data.columns
data.isnull().sum()
df=data.dropna()
df.shape
df['outcome'].unique()
df[data['outcome']==0].shape
df[data['outcome']==1].shape
df.tail()
x=df['Systolic blood pressure'].unique()
x.sort()
print(x)
df['Systolic blood pressure'].nunique()
print('Maximum Age :',df['age'].max())
```

```
print('Minimum Age :',df['age'].min())
print('Mean of Age :','%.2f%df['age'].mean())
print('Median of Age :','%.2f%df['age'].median())
print('Mode of Age :','%.2f%df['age'].mode())
df.describe()
df.corr()
df.info()
pd.crosstab(df['outcome'],df['age'])
pd.crosstab(df['outcome'],df['BMI'])
pd.crosstab(df['outcome'],df['heart rate'])
df.columns
df['PH'].value_counts()
pd.Categorical(df['age']).describe()
pd.Categorical(df['heart rate']).describe()
df.duplicated()
sum(df.duplicated())
```

VISUALIZATION

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('hospital.csv')
data
df=data.dropna()
df
df=df.rename({'heart rate': 'heart_rate','Systolic blood pressure' :'Systolic_blood_pressure'
```

```
        ,'Diastolic blood pressure':  'Diastolic_blood_pressure','Respiratory
rate':'Respiratory_rate',
         'SP O2': 'SP_O2','Urine output': 'Urine_output'},axis=1)
df.columns
pd.crosstab(df.heart_rate,df.temperature)
# Histogram
df['BMI'].hist(figsize=(10,5),color='y',alpha=1)
plt.xlabel('Minimum BMI')
plt.ylabel('Maximum BMI')
plt.title('BMI')
plt.figure(figsize=(8,4))
plt.hist(df['heart_rate'])
plt.title('Heart Rate')
plt.xlabel('Heart Rate Level')
plt.ylabel('Highest Heart Rate Range')
plt.show()
plt.figure(figsize=(9,6))
sns.scatterplot(x=df['age'],y=df['PH'], color='r')
plt.show()
plt.figure(figsize=(12,6))
plt.scatter(df['BMI'],df['glucose'],color='green')
plt.boxplot(df['BMI'])
plt.show()
sns.boxplot(df['heart_rate'],color='g')
plt.show()
sns.boxplot(df['BMI'],color='y')
plt.show()
plt.figure(figsize=(18,10))
plt.subplot(2,3,1)
plt.hist(df['Respiratory_rate'],color='violet')
plt.xlabel('Respiratory Rate')
plt.subplot(2,3,2)
plt.hist(df['PH'],color='chocolate')
plt.xlabel('PH')
plt.show()
fig, ax = plt.subplots(figsize=(24,7))
sns.stripplot(y=df['outcome'],x=df['age'],ax=ax)
plt.title('Age Wise Patient Results')
plt.show()
def outcome(df, variable):
    dataframe_pie=df[variable].value_counts()
    ax = dataframe_pie.plot.pie(figsize=(5,5),autopct='%1.2f%%', fontsize=10)
    ax.set_title(variable + ' \n', fontsize = 10)
    return np.round(dataframe_pie/df.shape[0]*100,2)
```

```
age(df, 'outcome')
fig,ax=plt.subplots(figsize=(25,12))
sns.heatmap(df.corr(),ax=ax,annot=True)
```

SUPPORT VECTOR MACHINE

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
# LOAD GIVEN DATASET
data=pd.read_csv('hospital.csv')
df=data.dropna()
df.columns
df.head()
df=df.rename({'heart rate':'heart_rate','Systolic blood pressure':'Systolic_blood_pressure',
        'Diastolic blood pressure':'Diastolic_blood_pressure','Respiratory
rate':'Respiratory_rate',
        'SP O2':'SP_O2','Urine output':'Urine_output'},axis=1)
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod=['outcome','age','BMI','heart_rate','Systolic_blood_pressure','Diastolic_blood_p
ressure',
      'Respiratory_rate','temperature','SP_O2','Urine_output','glucose','PH']
le=LabelEncoder()
for i in var_mod:
  df[i]=le.fit_transform(df[i]).astype(int)
df.head()
# preprocessing, split test and dataset, split response variable
X=df.drop(labels='outcome',axis=1)
# Response variable
y=df.loc[:,'outcome']
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X=scaler.fit_transform(X,y)
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
ros=RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print('OUR DATASET COUNT      :',Counter(y))
print('OVER SAMPLING DATA COUNT :',Counter(y_ros))
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x_ros,y_ros,test_size=0.30,random_state=4
2,stratify=y_ros)
```

```
print('Number of training dataset:',len(X_train))
print('Number of test dataset    :',len(X_test))
print('Total number of dataset   :',len(X_train)+len(X_test))
# Implementing SVC logo
from sklearn.svm import SVC
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score,plot_confusion_matrix
# Training
svc=SVC()
svc.fit(X_train,y_train)
predicted=svc.predict(X_test)
# Finding Accuracy
accuracy=accuracy_score(y_test,predicted)
print('Accuracy of Support Vector Classifier',accuracy*100)
# Finding Classification Report
cr=classification_report(y_test,predicted)
print('Classification report\n\n',cr)
# Finding Confusion Matrix
cm=confusion_matrix(y_test,predicted)
print('Confusion matrix\n\n',cm)
import matplotlib.pyplot as plt
fig,ax=plt.subplots(figsize=(10,10))
plot_confusion_matrix(svc,X_test,y_test,ax=ax)
plt.title('Confusion matrix of SVC')
plt.show()
df2=pd.DataFrame()
df2['y_test']=y_test
df2['predicted']=predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20,5))
plt.plot(df2['predicted'][:100],marker='x',linestyle='dashed',color='green')
plt.plot(df2['y_test'][:100],marker='o',linestyle='dashed',color='red')
plt.show()
```

K-NEAREST NEIGHBOR

```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
data=pd.read_csv('hospital.csv')
df=data.dropna()
df
df.columns
df=df.rename({'heart rate':'heart_rate','Systolic blood pressure':'Systolic_blood_pressure',
```

```
        'Diastolic blood pressure':'Diastolic_blood_pressure','Respiratory
rate':'Respiratory_rate',
        'SP O2':'SP_O2','Urine output':'Urine_output'},axis=1)
df.head()
from sklearn.preprocessing import LabelEncoder
knn=['outcome','age','BMI','heart_rate','Systolic_blood_pressure','Diastolic_blood_pressu
re',
    'Respiratory_rate','temperature','SP_O2','Urine_output','glucose']
label=LabelEncoder()
for i in knn:
   df[i]=label.fit_transform(df[i].astype(int))
df.head()
# Preprocessing, split test and dataset, split response variable
X=df.drop(labels='outcome',axis=1)
# Response variable
y=df['outcome']
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X=scaler.fit_transform(X,y)
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
ros=RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print('OUR DATASET COUNT       :',Counter(y))
print('OVER SAMPLING DATA COUNT :',Counter(y_ros))
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x_ros,y_ros,test_size=0.30,random_state=4
2,stratify=y_ros)
print("Number of training dataset :",len(X_train))
print('Number of test dataset     :',len(X_test))
print('Total Number of dataset    :',len(X_train)+len(X_test))
# Fitting KNN Classifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score,plot_confusion_matrix
# Training
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
predicted=knn.predict(X_test)
# Finding Accuracy
accuracy=accuracy_score(y_test,predicted)
print('Accuracy of KNN',accuracy*100)
# Finding Classification Report
```

```
cr=classification_report(y_test,predicted)
print('Classification report\n\n',cr)
# Finding Confusion Matrix
cm=confusion_matrix(y_test,predicted)
print('Confusion matrix \n\n',cm)
import matplotlib.pyplot as plt
fig,ax=plt.subplots(figsize=(10,10))
plot_confusion_matrix(knn,X_test,y_test,ax=ax)
plt.title('Confusion Matrix of KNN')
plt.show()
df2=pd.DataFrame()
df2['y_test']=y_test
df2['predicted']=predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20,5))
plt.plot(df2['predicted'][:100],marker='x',linestyle='dashed',color='violet')
plt.plot(df2['y_test'][:100],marker='o',linestyle='dashed',color='blue')
plt.show()
```

RANDOM FOREST CLASSIFIER

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
df=pd.read_csv('hospital.csv')
df.head()
df = df.dropna()
df.columns
df=df.rename({'heart rate':'heart_rate','Systolic blood pressure':'Systolic_blood_pressure',
        'Diastolic blood pressure':'Diastolic_blood_pressure','Respiratory
rate':'Respiratory_rate',
        'SP O2':'SP_O2','Urine output':'Urine_output'},axis=1)
df.columns
```

```
from sklearn.preprocessing import LabelEncoder
knn=['outcome','age','BMI','heart_rate','Systolic_blood_pressure','Diastolic_blood_pressu
re',
    'Respiratory_rate','temperature','SP_O2','Urine_output','glucose']
label=LabelEncoder()
for i in knn:
   df[i]=label.fit_transform(df[i].astype(int))
df['outcome'].unique()
df.head()
df.info()
# Preprocessing, split test and dataset, split response variable
X=df.drop(labels='outcome',axis=1)
# Response variable
y=df['outcome']
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X=scaler.fit_transform(X,y)
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter

ros=RandomOverSampler(random_state=42)
x_ros,y_ros=ros.fit_resample(X,y)
print('OUR DATASET COUNT       :',Counter(y))
print('OVER SAMPLING DATA COUNT :',Counter(y_ros))
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x_ros,y_ros,test_size=0.30,random_state=4
2,stratify=y_ros)
print("Number of training dataset :",len(X_train))
print('Number of test dataset     :',len(X_test))
```

```
print('Total Number of dataset  :',len(X_train)+len(X_test))
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
predicted = RFC.predict(X_test)
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF RANDOM FOREST CLASSIFIER:\n\n',cr)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n\n',cm)
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS :",a*100)
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS :",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
```

```
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
plt.plot(df2["y_test"][:100],  marker='o', linestyle='dashed', color='green')
plt.show()
import joblib
joblib.dump(RFC,'model.pkl')
```

A.3 Screen Shots

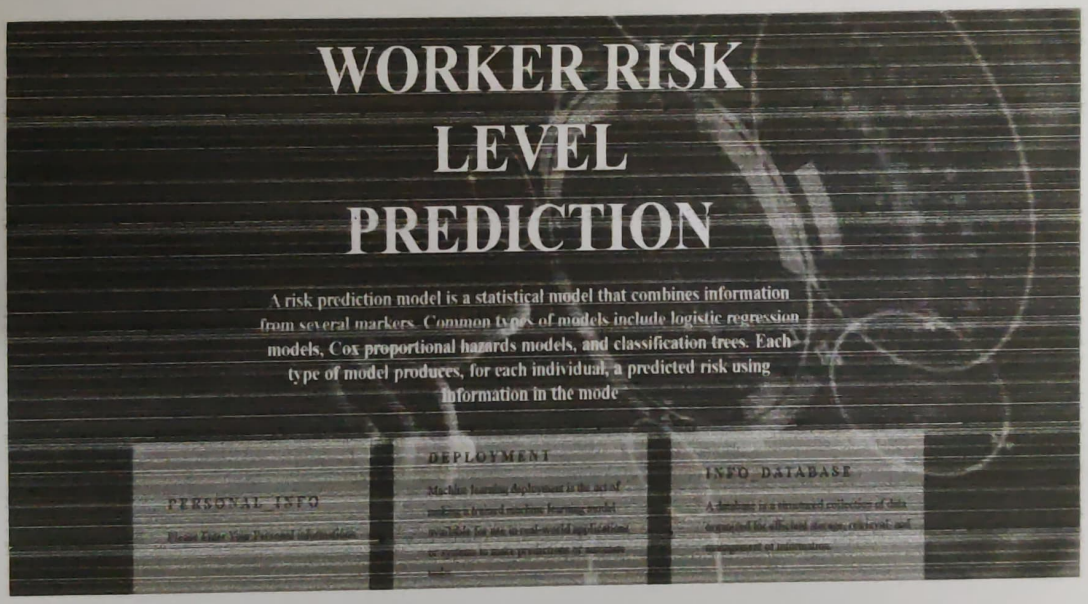

Fig A.1 Home page

Fig A.2 Registration page

Fig A.3 Login page