Configuration Manual

MSc Research Project
Data Analytics

Forename Surname
Student ID: x19213590

School of Computing
National College of Ireland

Supervisor: Dr. Majid Latifi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: …….Silky Jain……

Student ID: ……x19213590……..……

Programme: ……MSc in Data Analytics…… **Year:** ……2021……..

Module: ……MSc Research Project ……..……..

Lecturer: ……Dr. Majid Latifi……..……

Submission Due Date: ……23/09/2021……..……

Project Title: Prediction of Length of Stay and Hospital Readmission for Diabetic Patients……..……

Word Count: ……1405…… **Page Count:** ……14……..…..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Silky Jain……

Date: ……23/09/2021……..……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Silky Jain
Student ID: x19213590

1 Introduction

The purpose of building this document is to present the implementation of the project in a concise and structured manner such that it can be replicated if required. This project aimed to build a model for the prediction of length of stay and hospital readmission using a comprehensive methodological approach involving data cleaning and data pre-processing. Another objective of this project was to identify the most influencing factors for predicting length of stay and hospital readmission. The stacked ensemble learning model is proposed which is then compared with the other base classifiers and existing models on the same field. The tools and techniques used in the project are specified in this manual.

2 System Specifications

Below are the hardware requirements which is required for running the experiment and executing code smoothly.

Operating System	Windows 10 Home
RAM	8.0 GB
Har Disk Space	100 GB Minimum
Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz

3 Tools/Technology

For implementing this project, Python programming language was used with Integrated Development Environment (IDE) as Jupyter Notebook working on the Anaconda platform. The specific versions of the respective platform/language are mentioned below:

Programming Language	Python 3.8.3
IDE	Jupyter Notebook v. 6.0.3
Platform	Anaconda v. 4.9.2
Tools	Microsoft Excel, Overleaf, TeXstudio
Web Browser	Google Chrome

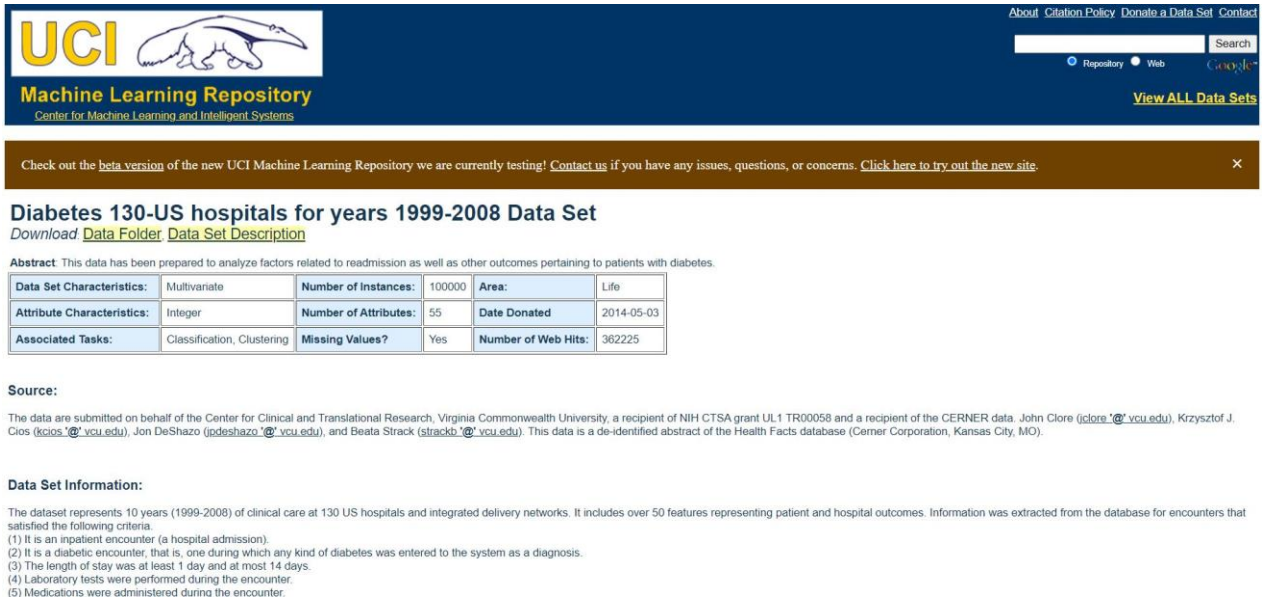
4 Pre-requisites software setup

The first step for the execution of this project is the installation of the required platform and languages.

- Python is installed using the link[1].
- Anaconda has been installed using this link[2].
- After execution of the project, the results are visualized in the Jupyter Notebook using the libraries such as MatPlotLib, seaborn, and Plotly.

5 Data Collection

Data for this project is extracted from the UCI Machine learning repository[3] and data description is in (Strack *et al.*, 2014).



6 Implementation

There are various processes involved in the implementation of this project which is given below:

6.1 Data Preparation and Storage

- Extracting the CSV file from the UCI machine learning repository to the local system.
- Importing the libraries in Jupyter Notebook is shown in Figure below.

[1] https://www.python.org/downloads/release/python-383/
[2] https://anaconda.org/conda-forge/conda/files?version=4.9.2
[3] https://archive.ics.uci.edu/ml/datasets/diabetes+130-us+hospitals+for+years+1999-2008

Importing Libraries

In [1]:
```
# Importing Libraries
import numpy as np
import pandas as pd

# Visualization
import missingno as msno
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

# Metrics
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.neighbors import LocalOutlierFactor
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score,f1_score,recall_score,mean_squared_error, r2_score, roc_auc_sco
from sklearn.metrics import classification_report
import sklearn.metrics as metrics
from sklearn.metrics import precision_recall_fscore_support

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from xgboost import XGBClassifier
```

- Reading data from CSV files and storing it in the form of a dataframe.

Reading data from csv files

In [3]:
```
data = pd.read_csv("C:/Users/silky/Desktop/DAPA - predictive analysis\dataset_diabetes\dataset_diabetes\diabetic_data.csv")
def display_all(data):
    with pd.option_context("display.max_row", 100, "display.max_columns", 100):
        display(data)
display_all(data.head())
IDs_mapping = pd.read_csv("C:/Users/silky/Desktop/DAPA - predictive analysis\dataset_diabetes\dataset_diabetes\IDs_mapping.c

display_all(IDs_mapping.head(67))
```

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	payer
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	25	1	1	
1	149190	55629189	Caucasian	Female	[10-20)	?	1	1	7	3	
2	64410	86047875	AfricanAmerican	Female	[20-30)	?	1	1	7	2	
3	500364	82442376	Caucasian	Male	[30-40)	?	1	1	7	2	
4	16680	42519267	Caucasian	Male	[40-50)	?	1	1	7	1	

	admission_type_id	description
0	1	Emergency

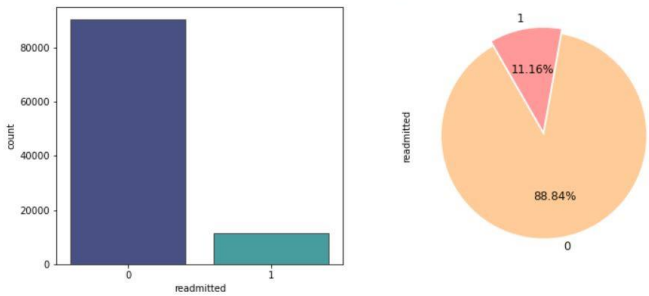
6.2 Exploratory Data Analysis

Before any cleaning and transformation of data, it is important to understand the data and to be aware of the features to focus on to undertake data cleaning and pre-processing.

Exploratory Data Analysis

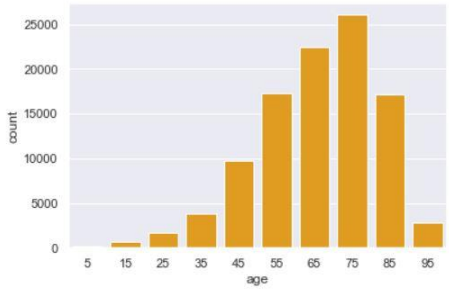
In [4]:
```
data.readmitted = [1 if each=='<30' else 0 for each in data.readmitted]
```

In [5]:
```
# Readmission data on plots
fig, ax =plt.subplots(nrows=1,ncols=2, figsize=(12,5))
labels=['0','1']
sns.countplot(x=data.readmitted, data=data, palette="mako",ax=ax[0], edgecolor=".3")
data.readmitted.value_counts().plot.pie(autopct="%1.2f%%", ax=ax[1], colors=['#ffcc99','#ff9999'],
                                        labels=labels, explode = (0, 0.05), startangle=120,
                                        textprops={'fontsize': 12, 'color':'#0a0a00'})
plt.show()
```

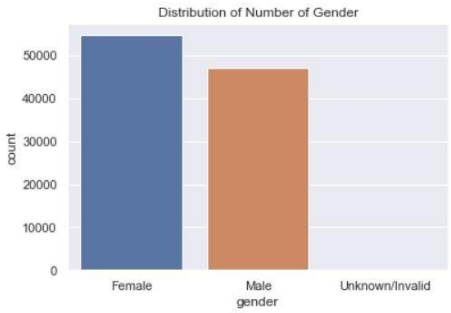


The above figure shows the plot showing the number of patients who were readmitted or not from the dataset.

In [62]:
```
diabet.age = diabet.age.replace({"[0-10)":5,
                                 "[70-80)":75,
                                 "[60-70)":65,
                                 "[50-60)":55,
                                 "[80-90)":85,
                                 "[40-50)":45,
                                 "[30-40)":35,
                                 "[90-100)":95,
                                 "[20-30)":25,
                                 "[10-20)":15}
                                 )

sns.countplot(x="age", data = diabet, color = 'orange')
#plt.xticks(rotation = 90)
plt.show()
```



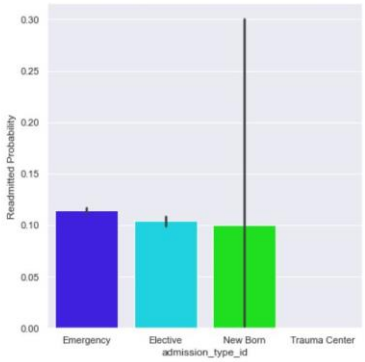
In [58]:
```
# Countplot for count of patients based on gender
sns.countplot(x = "gender", data = diabet)
plt.title("Distribution of Number of Gender")
plt.show()

print("Proportions of Race Value")
print(diabet.gender.value_counts(normalize = True))
```



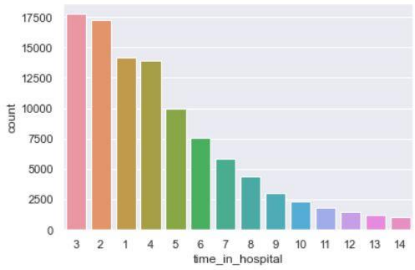

```
Proportions of Race Value
Female             0.537586
Male               0.462384
Unknown/Invalid    0.000029
Name: gender, dtype: float64
```

The figure shows the plot of the count of patients based on the gender and age group which shows that there is an almost equal number of patients in terms of gender and there are a greater number of patients observed from the age 65 to 85 years.

In [69]:
```
# Readmission probability vs admission type
g = sns.catplot(x = "admission_type_id", y ="readmitted",
                data = diabet, height = 6, kind = "bar", palette = 'gist_rainbow_r')
g.set_ylabels("Readmitted Probability")
plt.show()
```



In [74]:
```
# Count of patients spending time in hospital based on number of days
sns.countplot(x="time_in_hospital", data = diabet,
              order = diabet.time_in_hospital.value_counts().index)
plt.show()

print(diabet.time_in_hospital.value_counts())
```



These figures show the readmission of patients concerning admission type, so there are high cases of emergencies. It also shows the number of days patients stay at the hospital where most of the patients stayed for 1 to 3 days while admitted.

6.3 Data Pre-processing

Data cleaning is an important step in model building to get optimal performance. The steps involved in cleaning are converting the datatypes of the variables, checking and removal of duplicate rows, dropping columns with high multicollinearity, checking the columns having missing values and replacing them mean or other values, or dropping the columns if the percentage of the missing value is high as shown in Figures below.

Missing Values

In [99]:
```
#Checking for missing values in dataset
#In the dataset missing values are represented as '?' sign
for col in df.columns:
    if df[col].dtype == object:
         print(col,df[col][df[col] == '?'].count())
```

```
race 2273
gender 0
age 0
weight 98569
payer_code 40256
medical_specialty 49949
diag_1 21
diag_2 358
diag_3 1423
max_glu_serum 0
A1Cresult 0
metformin 0
repaglinide 0
nateglinide 0
chlorpropamide 0
glimepiride 0
acetohexamide 0
glipizide 0
glyburide 0
tolbutamide 0
```

In [100]:
```
# gender was coded differently so we use a custom count for this one
print('gender', df['gender'][df['gender'] == 'Unknown/Invalid'].count())
```

```
gender 3
```

In [101]:
```
#dropping columns with large number of missing values
df = df.drop(['weight','payer_code','medical_specialty'], axis = 1)
```

In [102]:
```
drop_Idx = set(df[(df['diag_1'] == '?') & (df['diag_2'] == '?') & (df['diag_3'] == '?')].index)

drop_Idx = drop_Idx.union(set(df['diag_1'][df['diag_1'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['diag_2'][df['diag_2'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['diag_3'][df['diag_3'] == '?'].index))
drop_Idx = drop_Idx.union(set(df['race'][df['race'] == '?'].index))
drop_Idx = drop_Idx.union(set(df[df['discharge_disposition_id'] == 11].index))
drop_Idx = drop_Idx.union(set(df['gender'][df['gender'] == 'Unknown/Invalid'].index))
new_Idx = list(set(df.index) - set(drop_Idx))
df = df.iloc[new_Idx]
```

In [103]:
```
df = df.drop(['citoglipton', 'examide'], axis = 1)
```

In [104]:
```
#Checking for missing values in the data
for col in df.columns:
    if df[col].dtype == object:
         print(col,df[col][df[col] == '?'].count())

print('gender', df['gender'][df['gender'] == 'Unknown/Invalid'].count())
```

```
race 0
gender 0
age 0
diag_1 0
diag_2 0
diag_3 0
max_glu_serum 0
A1Cresult 0
metformin 0
```

6.4 Data Transformation

After the data is cleaned there are steps to transform data so that it becomes easy for the machine learning algorithms to process it. The steps involved in pre-processing are feature engineering, feature encoding, feature selection, sampling, and scaling of data.

Encoding the outcome variable: The outcome we are looking at is whether the patient gets readmitted to the hospital within 30 days or not. The variable actually has < 30, > 30 and No Readmission categories. To reduce our problem to a binary classification, we combined the readmission after 30 days and no readmission into a single category

In [117]:
```
df['readmitted'] = df['readmitted'].replace('>30', 0)
df['readmitted'] = df['readmitted'].replace('<30', 1)
df['readmitted'] = df['readmitted'].replace('NO', 0)
```

In [118]:
```
# Creating additional columns for diagnosis
df['level1_diag1'] = df['diag_1']
df['level2_diag1'] = df['diag_1']
df['level1_diag2'] = df['diag_2']
df['level2_diag2'] = df['diag_2']
df['level1_diag3'] = df['diag_3']
df['level2_diag3'] = df['diag_3']
```

In [119]:
```
df.loc[df['diag_1'].str.contains('V'), ['level1_diag1', 'level2_diag1']] = 0
df.loc[df['diag_1'].str.contains('E'), ['level1_diag1', 'level2_diag1']] = 0
df.loc[df['diag_2'].str.contains('V'), ['level1_diag2', 'level2_diag2']] = 0
df.loc[df['diag_2'].str.contains('E'), ['level1_diag2', 'level2_diag2']] = 0
df.loc[df['diag_3'].str.contains('V'), ['level1_diag3', 'level2_diag3']] = 0
df.loc[df['diag_3'].str.contains('E'), ['level1_diag3', 'level2_diag3']] = 0
df['level1_diag1'] = df['level1_diag1'].replace('?', -1)
df['level2_diag1'] = df['level2_diag1'].replace('?', -1)
df['level1_diag2'] = df['level1_diag2'].replace('?', -1)
df['level2_diag2'] = df['level2_diag2'].replace('?', -1)
df['level1_diag3'] = df['level1_diag3'].replace('?', -1)
df['level2_diag3'] = df['level2_diag3'].replace('?', -1)
```

Feature encoding is done to encode the values of features into fewer categories such that it becomes easier for the ML algorithms to understand data as shown in the figure.

Data Transformation

Feature Engineering

Service Utilization

Service utilization: The data contains variables for number of inpatient (admissions), emergency room visits and outpatient visits for a given patient in the previous one year. These are (crude) measures of how much hospital/clinic services a person has used in the past year. We added these three to create a new variable called service utilization (see figure below).

In [107]:
```
df['service_utilization'] = df['number_outpatient'] + df['number_emergency'] + df['number_inpatient']
```

In [108]:
```
df.head(10).T
```

Out[108]:

	1	2	3	4	5	6	7	8	9	10
encounter_id	149190	64410	500364	16680	35754	55842	63768	12522	15738	28236
patient_nbr	55629189	86047875	82442376	42519267	82637451	84259809	114882984	48330783	63555939	89869032
race	Caucasian	AfricanAmerican	Caucasian	Caucasian	Caucasian	Caucasian	Caucasian	Caucasian	Caucasian	AfricanAmerican
gender	Female	Female	Male	Male	Male	Male	Male	Female	Female	Female
age	[10-20)	[20-30)	[30-40)	[40-50)	[50-60)	[60-70)	[70-80)	[80-90)	[90-100)	[40-50)
admission_type_id	1	1	1	1	2	3	1	2	3	1
discharge_disposition_id	1	1	1	1	1	1	1	1	3	1
admission_source_id	7	7	7	7	2	2	7	4	4	7
time_in_hospital	3	2	2	1	3	4	5	13	12	9
num_lab_procedures	59	11	44	51	31	70	73	68	33	47

The figure shows the engineering of the new feature ‘service_utilization’ by combining the existing feature in the dataset.

Sampling

In [160]:
```
# Importing libraries for SMOTE sampling
from imblearn.over_sampling import SMOTE
from collections import Counter
print('Original dataset shape {}'.format(Counter(y_train)))
```

```
Original dataset shape Counter({0: 43711, 1: 4053})
```

In [161]:
```
sm = SMOTE(random_state=20)
```

In [162]:
```
pip install --upgrade scikit-learn
```

```
Requirement already up-to-date: scikit-learn in c:\users\silky\anaconda3\lib\site-packages (0.24.2)
Requirement already satisfied, skipping upgrade: numpy>=1.13.3 in c:\users\silky\anaconda3\lib\site-packages (from scikit-lear
n) (1.19.5)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in c:\users\silky\anaconda3\lib\site-packages (from scikit-learn)
(0.16.0)
Requirement already satisfied, skipping upgrade: scipy>=0.19.1 in c:\users\silky\anaconda3\lib\site-packages (from scikit-lear
n) (1.5.0)
Requirement already satisfied, skipping upgrade: threadpoolctl>=2.0.0 in c:\users\silky\anaconda3\lib\site-packages (from sciki
t-learn) (2.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [163]:
```
df.head().T
```

Out[163]:

	1	2	3	4	5
encounter_id	149190	64410	500364	16680	35754
patient_nbr	55629189	86047875	82442376	42519267	82637451
race	Caucasian	AfricanAmerican	Caucasian	Caucasian	Caucasian
gender	0	0	1	1	1
age	2	3	4	5	6
admission_type_id	1	1	1	1	1

The figure shows the SMOTE oversampling where the target variable ‘readmitted’ is having imbalanced data which is balanced using the sampling technique.

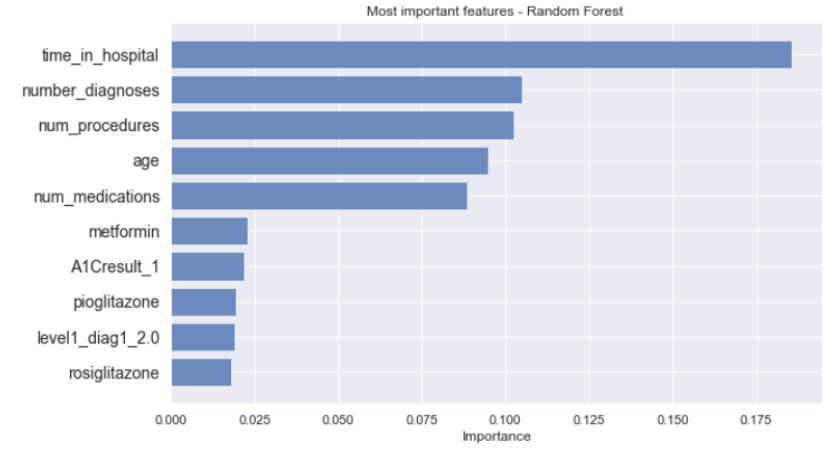
Features are selected based on the important feature identified by the Random Forest algorithm and then those features are pre-processed and used for building the model as shown in the figures below.

Feature Selection

In [124]:
```
# convert data type of nominal features in dataframe to 'object' type
i = ['encounter_id', 'patient_nbr', 'gender', 'admission_type_id', 'discharge_disposition_id', 'admission_source_id',\
        'A1Cresult', 'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide', 'glimepiride', 'acetohexamide', \
        'glipizide', 'glyburide', 'tolbutamide', 'pioglitazone', 'rosiglitazone', 'acarbose','miglitol', \
        'troglitazone', 'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin', \
        'glimepiride-pioglitazone', 'metformin-rosiglitazone', 'metformin-pioglitazone', 'change', 'diabetesMed', \
        'age', 'A1Cresult', 'max_glu_serum', 'level1_diag1', 'level1_diag2', 'level1_diag3', 'level2_diag1', 'level2_diag2

df[i] = df[i].astype('object')
```

In [125]:
```
df.dtypes
```

Out[125]:
```
encounter_id               object
patient_nbr                object
race                       object
gender                     object
age                        object
admission_type_id          object
discharge_disposition_id   object
admission_source_id        object
time_in_hospital            int64
num_lab_procedures          int64
num_procedures              int64
num_medications             int64
number_outpatient           int64
number_emergency            int64
number_inpatient            int64
diag_1                     object
diag_2                     object
```



Feature Scaling

In [141]:
```
# Feature Scaling
datf = pd.DataFrame()
datf['features'] = numerics
datf['std_dev'] = datf['features'].apply(lambda x: df[x].std())
datf['mean'] = datf['features'].apply(lambda x: df[x].mean())
```

In [142]:
```
# dropping multiple encounters while keeping either first or last encounter of these patients
df2 = df.drop_duplicates(subset= ['patient_nbr'], keep = 'first')
df2.shape
```

Out[142]: (67580, 55)

In [143]:
```
# standardize function
def standardize(raw_data):
    return ((raw_data - np.mean(raw_data, axis = 0)) / np.std(raw_data, axis = 0))
```

In [144]:
```
df2[numerics] = standardize(df2[numerics])
import scipy as sp
df2 = df2[(np.abs(sp.stats.zscore(df2[numerics])) < 3).all(axis=1)]
```

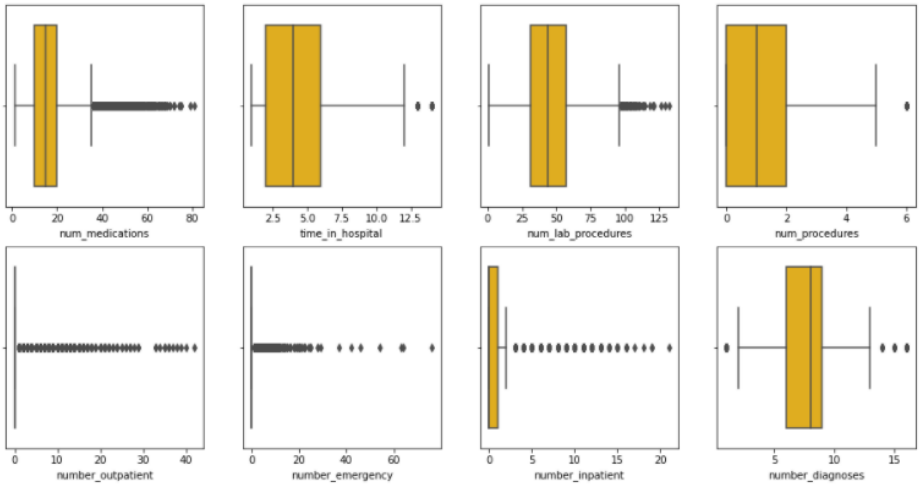
Feature Scaling is done to normalize or standardize the feature values in the attributes, so the results are optimized.

Outlier Detection

In [12]:
```
# Function for creating box plot for columns having outliers
def boxplot_for_outlier(df,columns):
    count = 0
    fig, ax =plt.subplots(nrows=2,ncols=4, figsize=(16,8))
    for i in range(2):
        for j in range(4):
            sns.boxplot(x = df[columns[count]], palette="Wistia",ax=ax[i][j])
            count = count+1
```

In [13]:
```
numerical_columns = ['num_medications',
  'time_in_hospital',
  'num_lab_procedures',
  'num_procedures',
  'number_outpatient',
  'number_emergency',
  'number_inpatient',
  'number_diagnoses']
```

In [14]:
```
boxplot_for_outlier(data,numerical_columns)
```



Outliers are detected and plotted in the form of a box plot and are removed as it influences the results giving biased predictions.

6.5 Data Modelling

The most important step of the data mining process is building a model. For building model, Scikit-learn machine learning library is used which have packages for data pre-processing, transformation, building model and evaluation metrics.

The base classifiers build for this project are – Random Forest, Decision Tree, k Nearest Neighbors, Support Vector Machines, Logistic Regression, Gradient Boosting and Extreme

Gradient Boosting. Then based on the performance of each classifier, the best models are selected for building a stacked ensemble model.

In [168]:
```
# Importing libraries
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
# Random Forest Classifier
rm = RandomForestClassifier(n_estimators = 10, max_depth=25, criterion = "gini", min_samples_split=10)
rm.fit(X_train, y_train)
```

Out[168]:
```
RandomForestClassifier(max_depth=25, min_samples_split=10, n_estimators=10)
```

In [169]:
```
# Prediction from random forest
rm_prd = rm.predict(X_test)
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(rm_prd, name = 'Predict'), margins = True)
```

Out[169]:

Predict	0	1	All
Actual			
0	4299	4555	8854
1	457	475	932
All	4756	5030	9786

In [170]:
```
# Accuracy, Precision and Recall from the RF classifier
print_report(y_test, rm_prd, thresh)
```

```
AUC:0.889
accuracy:0.889
recall:0.898
precision:0.882
fscore:0.890
specificity:0.879
```

This Figure shows the implementation of the Random Forest Classifier and the results obtained by using the testing data.

In [175]:
```
# Importing libraries
from sklearn.tree import DecisionTreeClassifier
# Decisison Tree Classifier
dtree = DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=28,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=10,
            min_weight_fraction_leaf=0.0, random_state=None,
            splitter='best')
dtree.fit(X_train, y_train)
```

Out[175]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=28, min_samples_split=10)
```

In [176]:
```
# Decision Tree prediction
dtree_pred = dtree.predict(X_test)
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(dtree_pred, name = 'Predict'), margins = True)
```

Out[176]:

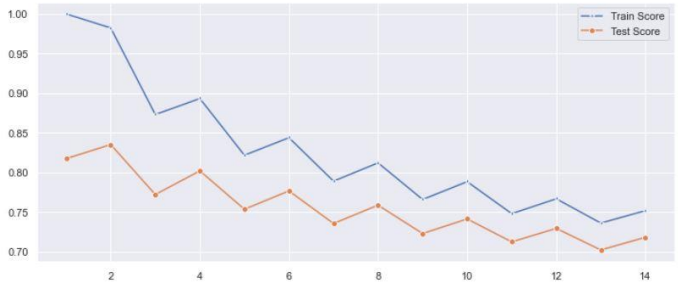
Predict	0	1	All
Actual			
0	8229	7538	15767
1	847	761	1608
All	9076	8299	17375

In [177]:
```
# Accuracy, precision and recall from Decision Tree Classifier
print_report(y_test, dtree_pred, thresh)
```

```
AUC:0.888
accuracy:0.888
recall:0.865
precision:0.906
fscore:0.885
specificity:0.910
```

This figure shows the implementation of the Decision Tree Classifier and the results obtained by using the testing data.

In [202]:
```
# Plotting the train and test data points
plt.figure(figsize=(12,5))
p = sns.lineplot(range(1,15),train_scores,marker='*',label='Train Score')
p = sns.lineplot(range(1,15),test_scores,marker='o',label='Test Score')
```



In [203]:
```
#Setup a knn classifier with k neighbors
knn = KNeighborsClassifier(2)

knn.fit(X_train,y_train)
knn.score(X_test,y_test)
```

Out[203]: 0.8350987165701997

This figure shows the implementation of k Nearest Neighbors, where the value of 'k' is selected optimally by implementing the values from 1 to 15 as shown in the plot and the results obtained by using the testing data while building a classifier for k=2.

SVM

SVM without hyperparameter

In [189]:
```
# import SVC classifier
from sklearn.svm import SVC


# instantiate classifier with default hyperparameters
svc=SVC()


# fit classifier to training set
svc.fit(X_train,y_train)


# make predictions on test set
y_pred=svc.predict(X_test)

print_report(y_test, y_pred, thresh)
```

```
AUC:0.764
accuracy:0.764
recall:0.781
precision:0.756
fscore:0.768
specificity:0.746
```

Out[189]:
```
(0.7635818715350413,
 0.7636521699343415,
 0.7809423988333941,
 0.7562331759410441,
 0.768389194036543,
 0.7462213442366885)
```

In [190]:
```
# SVM prediction
y_pred=svc.predict(X_test)
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(y_pred, name = 'Predict'), margins = True)
```

Out[190]:

Predict	0	1	All
Actual			
0	9595	10275	19870
1	832	988	1820
All	10427	11263	21690

This figure shows the implementation of the Support Vector Machine file and the results obtained by using the testing data, there were other experiments also done for other kernels like polynomial, sigmoid which is there in the code.

In [186]:
```
train_input_new = pd.DataFrame(train_input_new, columns = list(X.columns))
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
X_train, X_test, y_train, y_test = train_test_split(train_input_new, train_output_new, test_size=0.50, random_state=0)
# Logistic Regression with the L1 penalty
logit = LogisticRegression(fit_intercept=True, penalty='l1', solver='liblinear')
logit.fit(X_train, y_train)
```

Out[186]:
```
LogisticRegression(penalty='l1', solver='liblinear')
```

In [187]:
```
# Logistic Regression prediction
logit_pred2 = logit.predict(X_test)
pd.crosstab(pd.Series(y_test, name = 'Actual'), pd.Series(logit_pred2, name = 'Predict'), margins = True)
```

Out[187]:

Predict	0	1	All
Actual			
0	9666	10004	19670
1	856	964	1820
All	10722	10968	21690

In [188]:
```
print_report(y_test, logit_pred2, thresh)
```

```
AUC:0.745
accuracy:0.745
recall:0.747
precision:0.746
fscore:0.747
specificity:0.744
```

Out[188]:
```
(0.7453666196782243,
 0.7453730182333966,
 0.7469467736055414,
 0.7461307356154406,
 0.7465385316086719,
 0.7437864657509073)
```

This figure shows the implementation of Logistic Regression and the results obtained by using the testing data.

XG Boost

In [46]:
```
# XG Boost Classifier
xgb_model = XGBClassifier(random_state=42, n_jobs=-1,max_depth=3)
xgb_model.fit(X_train, y_train)
```

```
C:\Users\silky\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is de
precated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=Fa
lse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_
class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[00:17:42] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
```

Out[46]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=-1, num_parallel_tree=1, random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [47]:
```
# Predicting results from the XG Boost for training and validation datset
y_train_preds = xgb_model.predict_proba(X_train)[:,1]
y_val_preds = xgb_model.predict_proba(X_val)[:,1]

print("XGBOOST")
xgb_train_pred = gradient_model.predict_proba(X_train)[:,1]
xgb_val_pred = gradient_model.predict_proba(X_val)[:,1]

print("Gradient Boosing")
print('Training:')
xgb_train_auc, xgb_train_accuracy, xgb_train_recall, xgb_train_precision, xgb_train_fscore, xgb_train_specificity = print_re
print('Validation:')
xgb_val_auc, xgb_val_accuracy, xgb_val_recall, xgb_val_precision,xgb_val_fscore, xgb_val_specificity = print_report(y_val,xg
```

```
XGBOOST
Gradient Boosing
Training:
AUC:0.697
accuracy:0.641
recall:0.604
precision:0.649
fscore:0.626
specificity:0.678
```

This figure shows the implementation of Extreme Gradient Boosting and the results obtained by using the testing data.

Gradient Boost

In [44]:
```
# Gradient Boost model
gradient_model = GradientBoostingClassifier(random_state=42)
gradient_model.fit(X_train, y_train)
```

Out[44]:
```
GradientBoostingClassifier(random_state=42)
```

In [45]:
```
# Training and validation prediction from Gradient Boosting
gb_train_pred = gradient_model.predict_proba(X_train)[:,1]
gb_vald_pred = gradient_model.predict_proba(X_val)[:,1]

print("Gradient Boosing")
print('Training:')
gbc_train_auc, gbc_train_accuracy, gbc_train_recall, gbc_train_precision,gbc_train_fscore, gbc_train_specificity = print_rep
print('Validation:')
gbc_val_auc, gbc_val_accuracy, gbc_val_recall, gbc_val_precision, gbc_val_fscore, gbc_val_specificity = print_report(y_val,g
```

```
Gradient Boosing
Training:
AUC:0.697
accuracy:0.641
recall:0.604
precision:0.649
fscore:0.626
specificity:0.678

Validation:
AUC:0.663
accuracy:0.626
recall:0.595
precision:0.644
fscore:0.619
specificity:0.658
```

This figure shows the implementation of Gradient Boosting and the results obtained by using the testing data.

Then finally the stacked ensemble model is built by selecting only the best performing models and then the results of those base models are used to train the meta classifier Random Forest and final prediction results are obtained. The 'vecstack' package is used to stack together with the base models.

Stacked Ensemble Model - Final Model

In [206]:
```
# selecting list of top performing models to be used in stacked ensemble method
models = [
    GradientBoostingClassifier(random_state=42),
    XGBClassifier(random_state=42, n_jobs=-1,max_depth=3),
    SVC(kernel='poly', C=1.0),
    KNeighborsClassifier(2),
    DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=28,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=10,
            min_weight_fraction_leaf=0.0, random_state=None,
            splitter='best'),
    LogisticRegression(fit_intercept=True, penalty='none'),
    LogisticRegression(fit_intercept=True, penalty='l1', solver='liblinear')
]
```

In [207]:
```
!pip install vecstack
```

This figure shows the stacking of models for hospital readmission prediction.

Stacked Ensemble Model - Final Model

In [132]:
```
# selecting list of top performing models to be used in stacked ensemble method
models = [
    SVC(kernel='poly', C=1.0),
    KNeighborsClassifier(2),
    DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=28,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=10,
            min_weight_fraction_leaf=0.0, random_state=None,
            splitter='best'),
    LogisticRegression(fit_intercept=True, penalty='none'),
    LogisticRegression(fit_intercept=True, penalty='l1', solver='liblinear')
]
```

In [133]:
```
!pip install vecstack
```

This figure shows the stacking of models for the length of stay prediction.

In [209]:

```
# Training the Stacked Ensemble model with 5 cross vaidations for the list of models selected
S_train, S_test = stacking(models,
                           X_train, y_train, X_test,
                           regression=False,

                           mode='oof_pred_bag',

                           needs_proba=False,

                           save_dir=None,

                           metric=accuracy_score,

                           n_folds=5,

                           stratified=True,

                           shuffle=True,

                           random_state=0,

                           verbose=2)
```

```
task:         [classification]
n_classes:    [2]
metric:       [accuracy_score]
mode:         [oof_pred_bag]
n_models:     [7]

model  0:     [GradientBoostingClassifier]
    fold  0:  [0.88184834]
    fold  1:  [0.88458019]
    fold  2:  [0.87714482]
    fold  3:  [0.89212995]
    fold  4:  [0.88469458]
    ----
    MEAN:     [0.88407957] + [0.00486840]
```

Then the base models are trained using 5-fold cross-validation where the first four folds are used to train the data and the last fold is used to test the data and get the final predictions which are then augmented to the dataset. This augmented dataset is then used to train the meta classifier Random Forest which is shown in the below figure.

In [210]:

```
# initializing generalizer model i.e., RF classifier in our case
model = RandomForestClassifier(n_estimators = 10, max_depth=25, criterion = "gini", min_samples_split=10)

model = model.fit(S_train, y_train)
y_pred = model.predict(S_test)
print('Final prediction score: [%.8f]' % accuracy_score(y_test, y_pred))
```

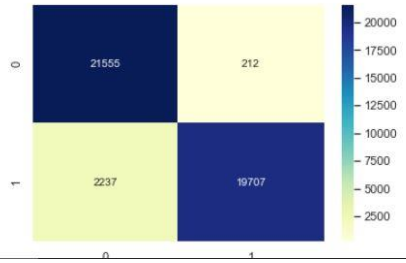
```
Final prediction score: [0.94397291]
```

In [211]:

```
CM=confusion_matrix(y_test,y_pred)
sns.heatmap(CM, annot=True,cmap="YlGnBu" ,fmt='g')

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
loss_log = log_loss(y_test, y_pred)
acc= accuracy_score(y_test, y_pred)
roc=roc_auc_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

mathew = matthews_corrcoef(y_test, y_pred)
model_results =pd.DataFrame([['STacked Classifier2',acc, prec,rec,specificity, f1,roc, loss_log,mathew]],
               columns = ['Model', 'Accuracy','Precision', 'Sensitivity','Specificity', 'F1 Score','ROC','Log_Loss','mathew_
model_results
```

Out[211]:

	Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log_Loss	mathew_corrcoef
0	STacked Classifier2	0.943973	0.989357	0.898059	0.99026	0.9415	0.94416	1.935111	0.89182



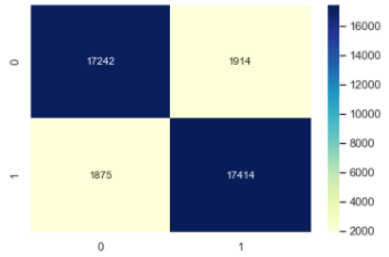
This figure shows the prediction results of the hospital readmission prediction using the Stacked Ensemble model.

In [139]:

```
# Results and confusion matrix for the stacked ensemble model
CM=confusion_matrix(y_test,y_pred)
sns.heatmap(CM, annot=True,cmap="YlGnBu" ,fmt='g')

TN = CM[0][0]
FN = CM[1][0]
TP = CM[1][1]
FP = CM[0][1]
specificity = TN/(TN+FP)
loss_log = log_loss(y_test, y_pred)
acc= accuracy_score(y_test, y_pred)
roc=roc_auc_score(y_test, y_pred)
prec = precision_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

mathew = matthews_corrcoef(y_test, y_pred)
model_results =pd.DataFrame([['STacked Classifier2',acc, prec,rec,specificity, f1,roc, loss_log,mathew]],
               columns = ['Model', 'Accuracy','Precision', 'Sensitivity','Specificity', 'F1 Score','ROC','Log_Loss','mathew_

model_results
```

Out[139]:

	Model	Accuracy	Precision	Sensitivity	Specificity	F1 Score	ROC	Log_Loss	mathew_corrcoef
0	STacked Classifier2	0.901444	0.900973	0.902794	0.900084	0.901883	0.901439	3.404057	0.802885



This figure shows the prediction results of the length of stay prediction using the Stacked Ensemble model.

7 Conclusion

All the pre-requisites – hardware and software configuration and requirement with the library and packages used for building models have been described in this document. This report explains the complete project development process in a structured, concise, and detailed manner which will help in understanding the flow of implementation.

References

Strack, B. *et al.* (2014) 'Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records', *BioMed Research International*. Edited by A. Rizvi, 2014, p. 781670. doi: 10.1155/2014/781670.