# Configuration Manual of Sepsis Prediction using Machine Learning and Deep Learning Algorithms

MSc Research Project
Data Analytics


Terrance Thomas
Student ID: X18184928

School of Computing
National College of Ireland


Supervisor: Dr. Rashmi Gupta

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Terrance Thomas |
| **Student ID:** | X18184928 |
| **Programme:** | Data Analytics |
| **Year:** | 2020 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Dr. Rashmi Gupta |
| **Submission Due Date:** | 17/08/2020 |
| **Project Title:** | Configuration Manual of Sepsis Prediction using Machine Learning and Deep Learning Algorithms |
| **Word Count:** | 1009 |
| **Page Count:** | 9 |

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:** | |
| **Date:** | 17th August 2020 |

**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 of Sepsis Prediction using Machine Learning and Deep Learning Algorithms

Terrance Thomas
X18184928

# 1 Introduction

The Configuration Manual gives a detailed description of the research project environment setup. This involves the configuration of the system, the source of the data set, the language used for programming, the description of libraries, the packages used and the code outputs.

This document also gives discussion of the results of various tests used in the research. This report also includes some relevant information, graphs, and output metrics that were a part of implementation. The reason for including such information in the Configuration Manual is that it is worth considering for relevance and discussion. For ease of understanding, the discussions in the research report with relevant information mentioned in the Configuration Manual have been appropriately referenced

# 2 Specification of Experimental Environment

## 2.1 System Specifications

The project was executed on a local system. The system configuration is as follows:

- **Intel i7 8th generation processor**
- **1TB hard drive**
- **16 GB RAM**
- **8 GB of Nvidia GTX 1080 graphic card**
- **Operating system: Windows 10**

## 2.2 Technical Specifications

### 2.2.1 Software

- **Python:** The software version of 3.7 has been used for the research. Most of the activities like data collection, cleaning, pre-processing, exploratory analysis, modelling and evaluation is done using the software

- **Anaconda-3 Jupyter Notebook:** Python 3.7 is supported by Anaconda-3, and is a scripting platform. Jupyter Notebook is a platform used to write the code addition to its potential to combine code and share it.
- **Microsoft Excel 2003:** Data is stored in .psv format which are pipe separated value files. There are 40,336 patient records/files. Each files contains hourly data of the patients.

### 2.2.2 Libraries

The following libraries were used for the research:

```
In [1]: import pandas as pd
from glob import glob
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import pickle
import os
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import time
from pylab import rcParams
import six
import sys
#sys.modules['sklearn.externals.six'] = six
import mlrose
from imblearn.under_sampling import NearMiss
import missingno as msno
from sklearn.metrics import classification_report, auc, precision_recall_curve, roc_curve
from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from collections import Counter
from keras import backend as K

Using TensorFlow backend.
C:\Users\Tero\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:143: FutureWarning: The sklearn.neighbors.base module is
deprecated in version 0.22 and will be removed in version 0.24. The corresponding classes / functions should instead be importe
d from sklearn.neighbors. Anything that cannot be imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
```

Figure 1: List of Libraries used

Figure 1 shows the list of libraries imported for the research project

# 3 Data set source

The data is used from a online challenge. The data part of an online challenge and is openly available on https://physionet.org/content/challenge-2019/1.0.0/ and clicking on access data to provide Name, Affiliation and email id to get the data which leads to a broken link but an alternate link is provided there which can be used to access the data or can use https://archive.physionet.org/users/shared/challenge-2019/ directly which is already bypassed.

| | HR | O2Sat | Temp | SBP | MAP | DBP | Resp | EtCO2 | BaseExcess | HCO3 | ... | Fibrinogen | Platelets | Age | Gender | Unit1 | Unit2 | HospAdmTime | ICULOS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 1 |
| 1 | 97.0 | 95.0 | NaN | 98.0 | 75.33 | NaN | 19.0 | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 2 |
| 2 | 89.0 | 99.0 | NaN | 122.0 | 86.00 | NaN | 22.0 | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 3 |
| 3 | 90.0 | 95.0 | NaN | NaN | NaN | NaN | 30.0 | NaN | 24.0 | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 4 |
| 4 | 103.0 | 88.5 | NaN | 122.0 | 91.33 | NaN | 24.5 | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 5 |
| 5 | 110.0 | 91.0 | NaN | NaN | NaN | NaN | 22.0 | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 6 |
| 6 | 108.0 | 92.0 | 36.11 | 123.0 | 77.00 | NaN | 29.0 | NaN | NaN | NaN | ... | NaN | NaN | 83.14 | 0 | NaN | NaN | -0.03 | 7 |

Figure 2: Data when loaded in python

Figure 2 shows few columns in the data set when loaded in python via a dataframe. Data set consist of 40,336 patients of 2 different hospitals in USA.

# 4 Data Exploratory Analysis and Pre-Processing

This section contains Data Exploratory Analysis (EDA) and Pre-Processing images which are important but were not included in the report.

```
In [2]: stock1 = sorted(glob('D:/Masters Data/RIC/Sepsis/training_setA/training/p0*.psv'))

In [3]: #List of psv files in the folder training_setA/training

        stock1

Out[3]: ['D:/Masters Data/RIC/Sepsis/training_setA/training\\p000001.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000002.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000003.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000004.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000005.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000006.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000007.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000008.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000009.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000010.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000011.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000012.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000013.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000014.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000015.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000016.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000017.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000018.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000019.psv',
         'D:/Masters Data/RIC/Sepsis/training_setA/training\\p000020.psv',

In [4]: #adding all the psv files in the folder to a single dataframe using the read_csv and delimiter seperator "|"

        df_A = pd.concat((pd.read_csv(file,sep='|' ).assign(filename = file)
                        for file in stock1), ignore_index = True)
```

Figure 3: Data set A when loaded

Figure 3 shows how the data file path are added stored in a variable and then how the files are loaded in a data frame. Data set B is loaded in a similar way and then both the data set are combined into a single data frame.

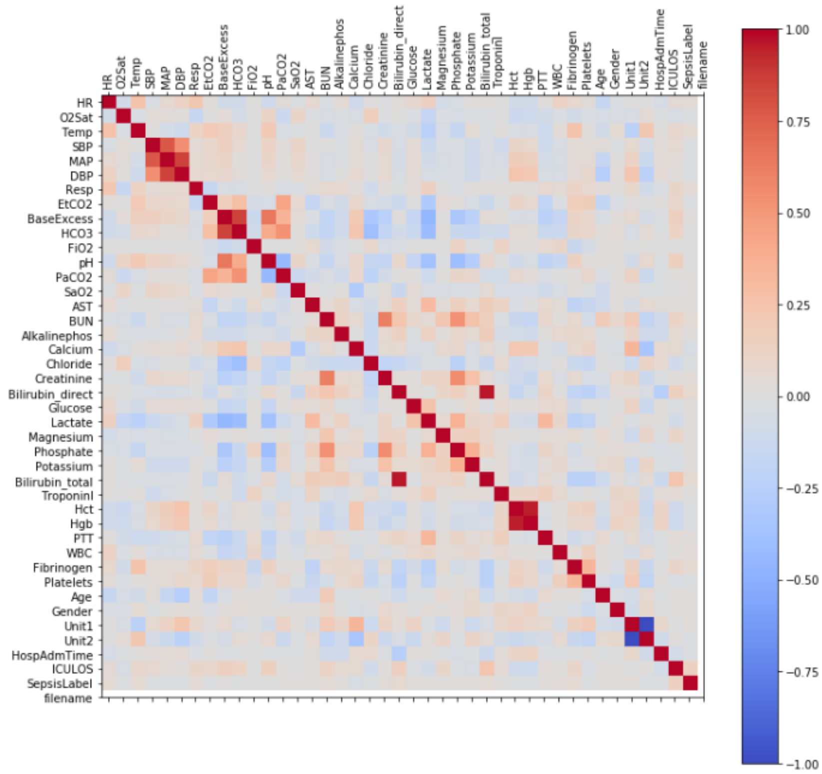


Figure 4: Correlation matrix of all columns

Figure 4 shows the correlation among all the columns/variables in the dataset.

```
In [30]: df_F.drop(['Temp','EtCO2','BaseExcess','HCO3','FiO2',
                    'pH','PaCO2','SaO2','AST','BUN','Alkalinephos','Calcium',
                    'Chloride','Creatinine','Bilirubin_direct','Glucose','Lactate',
                    'Magnesium','Phosphate','Potassium','Bilirubin_total','TroponinI',
                    'Hct','Hgb','PTT','WBC','Fibrinogen','Platelets','Unit1','Unit2'], axis=1, inplace=True)
```

Figure 5: Columns dropped

Figure 5 shows all the variables/columns which were dropped due to the high number of missing values.

```
In [101]: HR_median = df_F20['HR'].median()
          df_F20['HR'].fillna(HR_median,inplace=True)

          O2Sat_median = df_F20['O2Sat'].median()
          df_F20['O2Sat'].fillna(O2Sat_median,inplace=True)


          SBP_median = df_F20['SBP'].median()
          df_F20['SBP'].fillna(SBP_median,inplace=True)


          MAP_median = df_F20['MAP'].median()
          df_F20['MAP'].fillna(MAP_median,inplace=True)


          DBP_median = df_F20['DBP'].median()
          df_F20['DBP'].fillna(DBP_median,inplace=True)


          Resp_median = df_F20['Resp'].median()
          df_F20['Resp'].fillna(Resp_median,inplace=True)

          HospAdmTime_median = df_F20['HospAdmTime'].median()
          df_F20['HospAdmTime'].fillna(HospAdmTime_median,inplace=True)
```

Figure 6: Data Imputation

Figure 6 gives the detailed code of the imputation done on the data.

**Age**

```
In [60]: #Categorizing patient based on age to old, infant and Child/adult

def featuer_engineer_age(df_F10):
    df_F10.loc[df_F10['Age'] >=65, 'custom_age'] = 2
    df_F10.loc[df_F10['Age'] <1, 'custom_age'] = 0
    df_F10.loc[(df_F10['Age'] >=1) & (df_F10['Age'] <65),
            'custom_age'] = 1
    return df_F10
```

**O2Stat**

```
In [61]:
# O2Stat for a healthy adult is between 90 and 100 for healthy human beings. Create a new categorical variable custom_o2stat wit

def feature_engineer_o2stat(df_F10):
    df_F10.loc[(df_F10['O2Sat'] >= 90) & (df_F10['O2Sat'] < 100),
            'custom_o2stat'] = 1
    df_F10.loc[(df_F10['O2Sat'] < 90) & (df_F10['O2Sat'] >= 0),
            'custom_o2stat'] = 0

    df_F10['custom_o2stat'].fillna(np.nan, inplace=True)
    return df_F3
```

Figure 7: Data Imputation

Figure 7 shows a code for the feature extraction/encoding done on the data set.

```
In [69]: df_F10.isnull().sum()

Out[69]: Gender                0
HospAdmTime           0
ICULOS                0
SepsisLabel           0
custom_map            0
custom_bp       1151790
custom_hr             0
custom_o2stat    326414
custom_age            0
custom_resp           0
dtype: int64
```

Figure 8: Missing data after feature extraction

Figure 8 shows the number of data that was missing from the feature extractions. The missing values had to be imputed again.

```
In [78]: # Implementing Undersampling for Handling Imbalanced
nm = NearMiss(random_state=42)
X_res,y_res=nm.fit_sample(X,Y)

In [80]:
from collections import Counter
print('Original dataset shape {}'.format(Counter(Y)))
print('Resampled dataset shape {}'.format(Counter(y_res)))

Original dataset shape Counter({0: 1524294, 1: 27916})
Resampled dataset shape Counter({0: 27916, 1: 27916})
```

Figure 9: Under sampling

Figure 9 shows how NearMiss is used to convert the imbalanced data set into a balanced data set and matches the number of dependent variable.

```
In [119]: import matplotlib.pyplot as plt
          from sklearn.ensemble import RandomForestClassifier
          plt.figure(figsize=(20,10))

          model = RandomForestClassifier(random_state=1211)
          model.fit(x_train,y_train)
          # print(model.feature_importances_) #use inbuilt class feature_importances of tree based classifiers
          #plot graph of feature importances for better visualization
          feat_importances = pd.Series(model.feature_importances_, index=X.columns)
          feat_importances.nlargest(37).plot(kind='barh')
          plt.show()
```

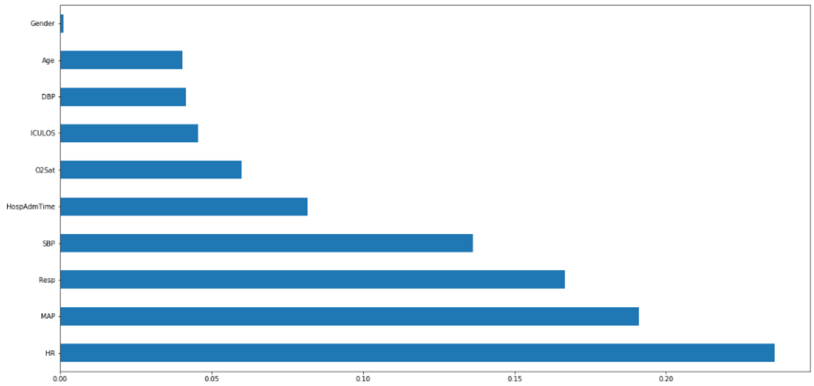


Figure 10: Variable importance

Figure 10 explains the variable importance of all the variables used in the data set after cleaning.

# 5 Models Implementation and Evaluation

This section shows all the model code and their output in terms of performance

## 5.1 Long Short Term Memory

```
In [91]: model = Sequential()
         model.add(LSTM(256, input_shape=(1,9), return_sequences = True))
         model.add(LSTM(256))
         model.add(Dense(128, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))
         model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy',f1_m,precision_m, recall_m])
         model.summary()

         Model: "sequential_3"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
         lstm_5 (LSTM)                (None, 1, 256)            272384
         _________________________________________________________________
         lstm_6 (LSTM)                (None, 256)               525312
         _________________________________________________________________
         dense_5 (Dense)              (None, 128)               32896
         _________________________________________________________________
         dense_6 (Dense)              (None, 1)                 129
         =================================================================
         Total params: 830,721
         Trainable params: 830,721
         Non-trainable params: 0
         _________________________________________________________________

In [95]: history = model.fit(X_train,y_train, epochs=50, batch_size=150, verbose=1, validation_data=(X_test,y_test), shuffle=True)

         Epoch 46/50
         41874/41874 [==============================] - 6s 132us/step - loss: 0.0656 - accuracy: 0.9184 - f1_m: 0.9128 - precision_m:
         0.9710 - recall_m: 0.8625 - val_loss: 0.0698 - val_accuracy: 0.9130 - val_f1_m: 0.9040 - val_precision_m: 0.9797 - val_recal
         l_m: 0.8413
         Epoch 47/50
         41874/41874 [==============================] - 5s 129us/step - loss: 0.0641 - accuracy: 0.9209 - f1_m: 0.9152 - precision_m:
         0.9740 - recall_m: 0.8641 - val_loss: 0.0682 - val_accuracy: 0.9156 - val_f1_m: 0.9098 - val_precision_m: 0.9646 - val_recal
         l_m: 0.8625
         Epoch 48/50
         41874/41874 [==============================] - 5s 127us/step - loss: 0.0639 - accuracy: 0.9209 - f1_m: 0.9155 - precision_m:
         0.9740 - recall_m: 0.8649 - val_loss: 0.0643 - val_accuracy: 0.9195 - val_f1_m: 0.9140 - val_precision_m: 0.9699 - val_recal
         l_m: 0.8657
         Epoch 49/50
         41874/41874 [==============================] - 5s 129us/step - loss: 0.0644 - accuracy: 0.9206 - f1_m: 0.9152 - precision_m:
         0.9736 - recall_m: 0.8644 - val_loss: 0.0656 - val_accuracy: 0.9196 - val_f1_m: 0.9137 - val_precision_m: 0.9736 - val_recal
         l_m: 0.8622
         Epoch 50/50
         41874/41874 [==============================] - 5s 131us/step - loss: 0.0652 - accuracy: 0.9195 - f1_m: 0.9141 - precision_m:
         0.9732 - recall_m: 0.8628 - val_loss: 0.0664 - val_accuracy: 0.9181 - val_f1_m: 0.9119 - val_precision_m: 0.9733 - val_recal
         l_m: 0.8593
```

Figure 11: Long Short Term Memory

## 5.2 Random Forest

```
In [165]: from sklearn.metrics import classification_report, auc, precision_recall_curve, roc_curve
          from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score

          best = RandomForestClassifier(random_state=1211, oob_score=True, max_depth=15, n_estimators=100, class_weight='balanced_subsampl
          best.fit(x_train,y_train)

          y_pred_best = best.predict(x_test)
          y_pred_proba_best = best.predict_proba(x_test)[::,1]
          precision_best, recall_best, thresholds_best = precision_recall_curve(y_test, y_pred_proba_best)
          auc_best = auc(recall_best, precision_best)
          fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba_best)
          auroc = auc(fpr, tpr)

          print(classification_report(y_test, y_pred_best))
          print('F1 score:', f1_score(y_test,y_pred_best))
          print('Precision:', precision_score(y_test,y_pred_best))
          print('Recall:', recall_score(y_test,y_pred_best))
          print('AUPRC:', auc_best)
          print('AUC:', auroc)
          print('Accuracy:', accuracy_score(y_test,y_pred_best))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      6930
           1       1.00      0.97      0.98      7028

    accuracy                           0.98     13958
   macro avg       0.98      0.98      0.98     13958
weighted avg       0.98      0.98      0.98     13958

F1 score: 0.9814319774582762
Precision: 0.9969176574196389
Recall: 0.9664200341491178
AUPRC: 0.993086791978159
AUC: 0.9891724793261504
Accuracy: 0.9815876200028657
```

Figure 12: Random Forest

## 5.3 Decision Tree

```
In [142]: clf_dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5,random_state=0)
          clf_dtc.fit(x_train, y_train.ravel())
          report_performance(clf_dtc)
          roc_curves(clf_dtc)
          accuracy(clf_dtc)
          f1(clf_dtc)
          precision(clf_dtc)
          recall(clf_dtc)
```

```
Acuuracy Of the Model:  0.9747814873191002


F1 score Of the Model:  0.974544402661267


Precision  Of the Model:  0.9908823529411764


Recall  Of the Model:  0.9587364826408651
```

Figure 13: Decision Tree

## 5.4 Extreme Gradient Boosting

```
In [144]: params = {
                'min_child_weight': [1, 5, 10],
                'gamma': [0.5, 1, 1.5, 2, 5],
                'subsample': [0.6, 0.8, 1.0],
                'colsample_bytree': [0.6, 0.8, 1.0],
                'max_depth': [1,2,3,4,5]
                }

          clf_xgb = XGBClassifier(random_state=0)
          clf_xgb.fit(x_train, y_train.ravel())
          report_performance(clf_xgb)
          roc_curves(clf_xgb)
          accuracy(clf_xgb)
          f1(clf_xgb)
          precision(clf_xgb)
          recall(clf_xgb)

          Acuuracy Of the Model:  0.9828055595357501


          F1 score Of the Model:  0.9826964671953857


          Precision  Of the Model:  0.996053785442853


          Recall  Of the Model:  0.9696926579396699
```

Figure 14: Extreme Gradient Boosting

## 5.5 Adaptive Boosting

```
In [145]: clfs =[AdaBoostClassifier()]

In [146]: for model in clfs:
              print(str(model).split('(')[0],": ")
              model.fit(x_train,y_train.ravel())
              X = pd.DataFrame(x_train)
              report_performance(model)
              roc_curves(model)
              accuracy(model)
              f1(model)
              precision(model)
              recall(model)

          Acuuracy Of the Model:  0.9771457228829346


          F1 score Of the Model:  0.9769591910436981


          Precision  Of the Model:  0.9920786269620068


          Recall  Of the Model:  0.9622936824132043
```

Figure 15: Adaptive Boosting

## 5.6 K-Nearest Neighbors

```
In [145]: clfs =[KNeighborsClassifier()]
```

```
In [146]: for model in clfs:
              print(str(model).split('(')[0],": ")
              model.fit(x_train,y_train.ravel())
              X = pd.DataFrame(x_train)
              report_performance(model)
              roc_curves(model)
              accuracy(model)
              f1(model)
              precision(model)
              recall(model)
```

```
Acuuracy Of the Model:  0.9727038257630033


F1 score Of the Model:  0.9723090340867795


Precision  Of the Model:  0.9937602139355222


Recall  Of the Model:  0.9517643710870802
```

Figure 16: K-Nearest Neighbors