***Machine Learning***

***on***

***USER KNOWLEDGE MODELLING***

***By,***

***Batch IV***

**Abstract**

Creating an efficient user knowledge model is a crucial task for web-based adaptive learning environments in different domains. It is often a challenge to determine exactly what type of domain dependent data will be stored and how it will be evaluated by a user modelling system. The most important disadvantage of these models is that they classify the knowledge of users without taking into account the weight differences among the domain dependent data of users. For this purpose, both the probabilistic and the instance-based models have been developed and commonly used in the user modelling systems. In this study a powerful, efficient and simple ‘Intuitive Knowledge Classifier’ method is proposed and presented to model the domain dependent data of users. A domain independent object model, the user modelling approach and the weight-tuning method are combined with instance-based classification algorithm to improve classification performances of well-known the Bayes and the k-nearest neighbour-based methods. The proposed knowledge classifier intuitively explores the optimum weight values of students’ features on their knowledge class first. Then it measures the distances among the students depending on their data and the values of weights. Finally, it uses the dissimilarities in the classification process to determine their knowledge class. The experimental studies have shown that the weighting of domain dependent data of students and combination of user modelling algorithms and population-based searching approach play an essential role in classifying performance of user modelling system. The proposed system improves the classification accuracy of instance-based user modelling approach for all distance metrics and different k-values.

**INDEX**

|  |  |  |
| --- | --- | --- |
|  | **Content** | *Pg no.* |
| 01 | Introduction – Problem Statement | 1 |
| 02 | Dataset Description | 2 |
| 03 | Architecture for Problem Statement | 3 |
| 04 | Machine Learning model | 4 |
| 05 | Results and Evaluation | 5-7 |
| 06 | Reference | 8 |
|  | *Appendix* |  |

**01 – Introduction**

**Problem Statement:**

Knowledge level of a user is to be predicted using the attributes

* The Degree of study time for goal object materials
* The Degree of repetition number of user for goal object materials
* The Degree of study time of user for related objects with goal object
* The Exam performance of user for related objects with goal object
* The Exam performance of user for goal objects

**02-Dataset Description**

**Attributes:**

* STG (The degree of study time for goal object materials), (input value)
* SCG (The degree of repetition number of user for goal object materials) (input value)
* STR (The degree of study time of user for related objects with goal object) (input value)
* LPR (The exam performance of user for related objects with goal object) (input value)
* PEG (The exam performance of user for goal objects) (input value)
* UNS (The knowledge level of user) (target value)
  + Very Low: 50
  + Low:129
  + Middle: 122
  + High 130

**03-Architecture of Problem Statement**

**Steps for solving problem:**

* Import required libraries
* Read the data from the database
* Correlate the data
* Split the data for training and testing
* Visualize data
* Transform the data into Normalised form or Standard Scaled Form
* Build Machine Learning model using suitable Regression or Classification techniques
* Evaluate the model and build Confusion matrix and find Classification report

**04-Machine Learning Model**

**Predicted values according to**

* Logistic Regression:

3 1 3 1 1 2 1 3 2 2 2 3 1 1 3 2 3 1 3 3 1 1 1 1 2 2 1 3 2 1 1 2 2 3 1 3 1 1 3 0 1 2 1 1 1 2 3 3 3 1 1 1

* K Nearest Neighbor:

2 1 3 2 2 1 1 3 2 2 2 2 1 0 3 2 3 1 3 3 0 2 1 1 1 2 1 3 2 1 0 2 2 3 2 2 2 1 3 0 1 2 1 1 1 1 1 2 3 2 1 1

* Support Vector Machine:

3 1 3 2 2 2 1 3 2 2 2 2 2 1 3 2 3 1 3 3 1 1 1 1 2 2 1 3 2 1 1 2 2 3 2 3 2 1 3 0 1 2 1 1 2 2 2 2 3 2 1 1

* Decision Tree:

3 1 3 2 1 2 0 3 2 2 2 2 2 0 3 2 3 2 3 3 0 1 0 1 2 2 1 3 2 1 1 2 2 3 2 3 1 1 3 0 0 2 1 0 2 2 3 2 3 1 1 2

* Random Forest:

3 1 3 2 2 2 1 3 2 2 2 2 2 1 3 2 3 1 3 3 0 1 1 1 2 2 1 3 2 1 1 2 2 3 2 3 1 1 3 0 1 2 1 0 2 2 3 2 3 1 1

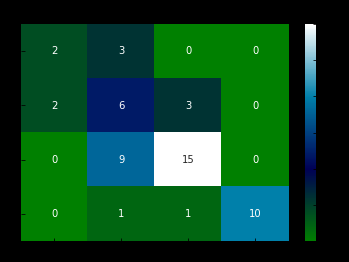
**05-Results and Evaluation**

**Confusion Matrix:**

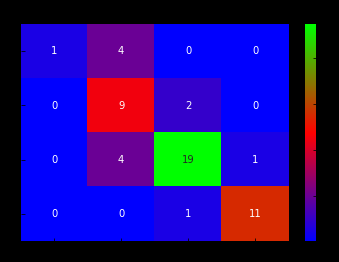
Logistic Regression:



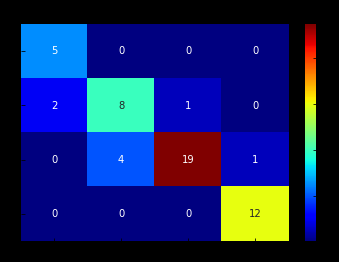
K nearest Neighbor



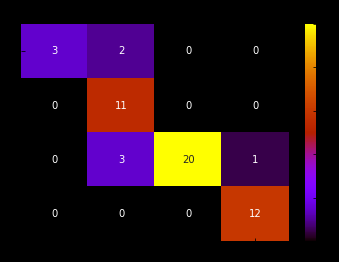
Support Vector Machine



Decision Tree



Random Forest Tree



**Classification Report:**

**Logistic Regression**

precision recall f1-score support

0 1.00 0.20 0.33 5

1 0.46 1.00 0.63 11

2 1.00 0.50 0.67 24

3 0.80 1.00 0.89 12

avg / total 0.84 0.69 0.68 52

**K Nearest Neighbor**

precision recall f1-score support

0 0.50 0.40 0.44 5

1 0.32 0.55 0.40 11

2 0.79 0.62 0.70 24

3 1.00 0.83 0.91 12

avg / total 0.71 0.63 0.66 52

**Support Vector Machine**

precision recall f1-score support

0 1.00 0.20 0.33 5

1 0.53 0.82 0.64 11

2 0.86 0.79 0.83 24

3 0.92 0.92 0.92 12

avg / total 0.82 0.77 0.76 52

**Decision tree**

precision recall f1-score support

0 0.71 1.00 0.83 5

1 0.67 0.73 0.70 11

2 0.95 0.79 0.86 24

3 0.92 1.00 0.96 12

avg / total 0.86 0.85 0.85 52

**Random Forest**

precision recall f1-score support

0 1.00 0.60 0.75 5

1 0.69 1.00 0.81 11

2 1.00 0.83 0.91 24

3 0.92 1.00 0.96 12

avg / total 0.92 0.88 0.89 52

**06-Reference**

**Link:**

[**https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling**](https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling)

***Appendix***

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import cv2

dataset=pd.read\_eximport numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import cv2cel('Book1.xlsx')

dataset.columns=['STG','SCG','STR','LPR','PEG','UNS']

dataset.head()

dataset.info()

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

dataset['UNS']=le.fit\_transform(dataset['UNS'])

dataset.head()

x=dataset.iloc[:,:-1].values # Independant variables

y=dataset.iloc[:,-1].values #dependant variables

x.shape,y.shape

plt.figure(figsize=(10,4))

plt.boxplot(x,vert =False,labels=['STG','SCG','STR','LPR','PEG'],

patch\_artist=True)

plt.show()

from sklearn.preprocessing import StandardScaler,MinMaxScaler

sc=StandardScaler() #z-score

mms=MinMaxScaler() #(0-1)->normalization

x\_sc =sc.fit\_transform(x)

x\_norm=mms.fit\_transform(x)

fig=plt.figure(figsize=(10,4))

plt.style.use('bmh')

# Without scaling

plt.boxplot(x,vert=False,labels=['STG','SCG','STR','LPR','PEG'],patch\_artist=True)

plt.title('Without Scaling')

plt.show()

# Normalisation

fig=plt.figure(figsize=(10,4))

plt.boxplot(x\_norm,vert=False,labels=['STG','SCG','STR','LPR','PEG'],patch\_artist=True)

plt.title('Normalisation(0-1)')

plt.show()

# Standard scaling

fig=plt.figure(figsize=(10,4))

plt.boxplot(x\_sc,vert=False,labels=['STG','SCG','STR','LPR','PEG'],patch\_artist=True)

plt.title('Standard Scaling(Z-score)')

plt.show()

from sklearn.cross\_validation import train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x\_sc,y,test\_size=0.2,random\_state=0)

x\_train.shape,y\_train.shape,x\_test.shape,y\_test.shape

cm\_log= confusion\_matrix(y\_test,y\_pred\_log)

cm\_knn= confusion\_matrix(y\_test,y\_pred\_knn)

cm\_svm= confusion\_matrix(y\_test,y\_pred\_svm)

cm\_dt= confusion\_matrix(y\_test,y\_pred\_dt)

cm\_rf= confusion\_matrix(y\_test,y\_pred\_rf)

print("\*"\*20+'Logistic Regression'+"\*"\*20)

print(cr\_log)

print("\*"\*20+'K Nearest Neighbor'+"\*"\*20)

print(cr\_knn)

print("\*"\*20+'Support Vector Machine'+"\*"\*20)

print(cr\_svm)

print("\*"\*20+'Decision tree'+"\*"\*20)

print(cr\_dt)

print("\*"\*20+'Random Forest'+"\*"\*20)

print(cr\_rf)