CO 544 - Machine Learning and Data Mining

Project Contribution Report

Group 15 – E/16/039

E/16/039

Balasuriya B.M.N.U

Analysis of Student Performance

Abstract

The object of this is to analyse and evaluate the university students performance of the common entrance examination (CEE), who qualified the medical entrance examination for admission to a medical college by applying different data mining and classification techniques using WEKA tool and python. The highest accuracy of classifier algorithm depends on the size and the nature of the data

Introduction

Educational data mining (EDM) is a very important research area which helpful to predict useful information from educational database to improve educational performance, better understanding and to have better assessment of the students learning process. Research on that will be helped to the students and the teachers to improve the result of the students who are at the risk of failure. In this we analyse

Problem Description

- ### Data set

The dataset contains data of the candidates who qualified the medical entrance examination for admission to medical colleges of Assam of a particular year and collected by Prof. Jiten Hazarika.

Provided by : Dr. Sadiq Hussain, Dibrugarh University, sadiq '@' dibru.ac.in

Repository: https://archive.ics.uci.edu/ml/datasets/Student+Performance+on+an+entrance+examination

Number of instances: 666

Number of attributes: 12

All the data in the dataset are categorical and no null value can be observed.

Inputs: All the attributes exclude Performance

Output: Performance

- **Methodology**

The dataset is partitioned as training (80%) and testing(20%) dataset and the classifiers are trained using the trained data set. The correctness of the classifier is tested using test dataset.

Contribution

- Used isnull() to check the null values of the data set and there was no any null observed.

```
In [3]: df.isnull().values.any()
```

```
Out[3]: False
```

- As the data preprocessing encoded the categorical labels in to numerical values in the data set using LabelEncoder() in python. LabelEnocoder encode labels with a value between 0 and numberOfClasses – 1

```
In [4]: from sklearn.preprocessing import LabelEncoder


le = LabelEncoder()
df.Performance = le.fit_transform(df['Performance'])
df.Gender = le.fit_transform(df['Gender'])
df.Caste = le.fit_transform(df['Caste'])
df.coaching = le.fit_transform(df['coaching'])
df.time = le.fit_transform(df['time'])
df.Class_ten_education= le.fit_transform(df['Class_ten_education'])
df.twelve_education = le.fit_transform(df['twelve_education'])
df.medium = le.fit_transform(df['medium'])
df.Class_X_Percentage = le.fit_transform(df['Class_X_Percentage'])
df.Class_XII_Percentage = le.fit_transform(df['Class_XII_Percentage'])
df.Father_occupation = le.fit_transform(df['Father_occupation'])
df.Mother_occupation = le.fit_transform(df['Mother_occupation'])
#df.Gender = le.fit_transform(df['Gender'])

df
```

Out[4]:

	Performance	Gender	Caste	coaching	time	Class_ten_education	twelve_education	medium
0	1	1	0	0	2	2	0	1
1	1	1	1	2	5	2	0	2
2	1	1	1	1	5	1	1	1
3	1	1	0	2	2	2	0	2
4	1	1	0	1	5	2	1	1
...	...	...	...	...	...	...	...	...
661	0	0	3	2	2	2	0	1
662	0	1	3	2	4	2	0	1
663	0	1	3	2	5	2	1	1
664	0	1	3	2	4	2	0	1

- Without having any data filtering for data preprocessing the accuracy of the prediction has been checked using 3 classifiers
    - K-Nearest Neighbors – Using 10 neighbors
        - Accuracy: 0.53731343

```
In [44]: KNN_model = KNeighborsClassifier(n_neighbors=10)
         KNN_model.fit(X_train, y_train)
         KNN_prediction = KNN_model.predict(X_test)

         from sklearn.metrics import accuracy_score
         from sklearn.metrics import confusion_matrix

         print('accuracy of KNN ')
         print(accuracy_score(KNN_prediction, y_test))

         accuracy of KNN
         0.5373134328358209
```

    - SVM
        - Accuracy: 0.477611940
    - SVC
        - Accuracy: 0.50

```
In [45]: from sklearn.svm import SVC
         SVC_model = SVC()
         SVC_model.fit(X_train, y_train)
         SVC_prediction = SVC_model.predict(X_test)
         print(str(accuracy_score(SVC_prediction, y_test)))

         0.47761194029850745
```

```
In [25]: from sklearn import svm, datasets
         import sklearn.model_selection as model_selection
         from sklearn.metrics import accuracy_score
         from sklearn.metrics import f1_score
```

```
In [27]: rbf = svm.SVC(kernel='rbf', gamma=0.5, C=0.1).fit(X_train, y_train)
         poly = svm.SVC(kernel='poly', degree=3, C=1).fit(X_train, y_train)
         poly_pred = poly.predict(X_test)
         rbf_pred = rbf.predict(X_test)
```

```
In [28]: poly_accuracy = accuracy_score(y_test, poly_pred)
         poly_f1 = f1_score(y_test, poly_pred, average='weighted')
         print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
         print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))

         Accuracy (Polynomial Kernel):  50.00
         F1 (Polynomial Kernel):  47.61
```

- Therefore it observed that K Nearest Neighbors classifier gives more accuracy than SVC. Therefore it indicates that the dataset we used is not an easily separable using decision planes. Because the basic SVM uses linear hyper planes to separate classes and if we provide a different kernel then it will change.
  Also without any data preprocessing there, we didn't observe very good accuracy. So we use WEKA tool for the data preprocessing.

- First we filtered the data set according to the their correlation over performance

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 1 Performance):
        Correlation Ranking Filter
Ranked attributes:
 0.2173    3 Caste
 0.108    10 Class_XII_Percentage
 0.0957    9 Class_ X_Percentage
 0.0518    4 coaching
 0.0504    6 Class_ten_education
 0.0462    8 medium
 0.0382    5 time
 0.0372   11 Father_occupation
 0.0319    2 Gender
 0.0292   12 Mother_occupation
 0.028     7 twelve_education

Selected attributes: 3,10,9,4,6,8,5,11,2,12,7 : 11
```

We take 0.05 as threshold value and eliminate other attributes which have the correlation below the threshold value. After that used two classifiers to observe the accuracy of the data prediction.
Eliminated classes – medium, time, father occupation, mother occupation, twelve education, gender

  - Meta.multiclass Classifier
    - A metaclassifier for handling multi-class datasets with 2-class classifiers. This classifier is also capable of applying error correcting output codes for increased accuracy. If the base classifier cannot handle instance weights, and the instance weights are not uniform, the data will be resampled with replacement based on the weights before being passed to the base classifier.
      Accuracy: 54.2042%

```
Correctly Classified Instances         361               54.2042 %
Incorrectly Classified Instances       305               45.7958 %
```

- NaiveBayes
  - Class for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an UpdateableClassifier (which in typical usage are initialized with zero training instances) -- if we need the UpdateableClassifier functionality, we need to use the NaiveBayesUpdateable classifier. The NaiveBayesUpdateable classifier will use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances.
    Accuracy: 53.6036

Correctly Classified Instances	357	53.6036 %
Incorrectly Classified Instances	309	46.3964 %

Since both classifiers not giving much high accuracy we tried another method to select best suitable classes to give us the best prediction accuracy.

Therefore as the next step some other features are dropped by doing learner based feature selection.**Caste,coaching** and **class_ten_eduaction** are selected from it.

- Meta.multiclass Classifier

Correctly Classified Instances	363	54.5045 %
Incorrectly Classified Instances	303	45.4955 %

- Nave Bayes

Correctly Classified Instances	364	54.6547 %
Incorrectly Classified Instances	302	45.3453 %

Therefore we can observe that the above classifiers predict almost same accuracy. Because of that, we identified that we need to try more classifiers. The other classifiers were examined by the two of other members of our group.

References

1. Label Encoding – https://towardsdatascience.com/choosing-the-right-encoding-method-label-vs-onehot-encoder-a4434493149b#:~:text=LabelEncoder%20encode%20labels%20with%20a,value%20to%20as%20assigned%20earlier.&text=The%20categorical%20values%20have%20been,all%20label%20encoding%20is%20about

2. KNN over SVM – https://www.researchgate.net/post/The_accuracy_of_k-NN_is_greater_than_SVM_What_would_be_the_main_reason#:~:text=As%20KNN%20works%20better%20than,manifold%20that%20can%20be%20used

3. CorrelationAttributeEval – http://infochim.u-strasbg.fr/cgi-bin/weka-3-9-1/doc/weka/attributeSelection/CorrelationAttributeEval.html

4. Feature selection in weka – https://www.tutorialspoint.com/weka/weka_feature_selection.html