Report On Project Higher Education Students Performance Evaluation Dataset

INT-354

Submitted By : G. Gopi Krishna

Reg No : 12115851

Section : KM054

Roll No : RKM054A33

Submitted To : **Mrs. Jaspreet Kaur**

Introduction

The collection contains profiles of more than 1,000 kids, together with details about their age, gender, family history, prior academic accomplishments, and test results. Researchers and educators may utilize this data to better understand the elements that affect college performance and develop initiatives to enhance student outcomes.

Predictive modelling, clustering, classification, study of educational policy, and programme assessment are just a few uses for this data. The UCI link Machine Learning Repository website, which also offers access to a wide of selection of datasets for study and analysis, allows users to download the data.

Dataset Used

This dataset was produced in order to assess student performance in higher education in Portugal. It includes information on a range of student characteristics, including demographics, high school education, and academic achievement. The dataset also contains details about the social and economic circumstances of the pupils, including their parents' occupations, educational backgrounds, and standard of living.

Researchers have developed and evaluated a variety of the machine learning models using this dataset for forecasting student performance, identifying the critical variables that influence academic performance, and comprehending the association between students' academic performance and their socio - economic background.

Libraries Used

- **NumPy: -** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **Pandas:** Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.
- **Matplotlib:** It is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.
- **Seaborn:** Seaborn is a Python data visualization library based on . It provides a high-level interface for drawing attractive and informative statistical graphics.

Import :-

```
import numpy as np # for linear algebra
import pandas as pd # for data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt #for graphs
import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier as dtc
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_classif
```

Program Code:

Data Preprocessing :

```
df = pd.read_csv("student_prediction.csv") #loading the data
df   #printing the df
```

	STUDENTID	AGE	GENDER	HS_TYPE	SCHOLARSHIP	WORK	ACTIVITY	PARTNER	SALARY	TRANSPORT	...	PREP_
0	STUDENT1	2	2	3	3	1	2	2	1	1	...	
1	STUDENT2	2	2	3	3	1	2	2	1	1	...	
2	STUDENT3	2	2	2	3	2	2	2	2	4	...	
3	STUDENT4	1	1	1	3	1	2	1	2	1	...	
4	STUDENT5	2	2	1	3	2	2	1	3	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
140	STUDENT141	2	1	2	3	1	1	2	1	1	...	
141	STUDENT142	1	1	2	4	2	2	2	1	4	...	
142	STUDENT143	1	1	1	4	2	2	2	1	1	...	
143	STUDENT144	2	1	2	4	1	1	1	5	2	...	
144	STUDENT145	1	1	1	5	2	2	2	3	1	...	

145 rows × 33 columns

df.head() #prints the top 5 rows elements present in it.

	STUDENTID	AGE	GENDER	HS_TYPE	SCHOLARSHIP	WORK	ACTIVITY	PARTNER	SALARY	TRANSPORT	...	PREP_STU
0	STUDENT1	2	2	3	3	1	2	2	1	1	...	
1	STUDENT2	2	2	3	3	1	2	2	1	1	...	
2	STUDENT3	2	2	2	3	2	2	2	2	4	...	
3	STUDENT4	1	1	1	3	1	2	1	2	1	...	
4	STUDENT5	2	2	1	3	2	2	1	3	1	...	

5 rows × 33 columns

df.tail() #prints last 5 rows elements which is present in dataset.

	STUDENTID	AGE	GENDER	HS_TYPE	SCHOLARSHIP	WORK	ACTIVITY	PARTNER	SALARY	TRANSPORT	...	PREP_
140	STUDENT141	2	1	2	3	1	1	2	1	1	...	
141	STUDENT142	1	1	2	4	2	2	2	1	4	...	
142	STUDENT143	1	1	1	4	2	2	2	1	1	...	
143	STUDENT144	2	1	2	4	1	1	1	5	2	...	
144	STUDENT145	1	1	1	5	2	2	2	3	1	...	

5 rows × 33 columns

df.shape #prints in the format of (rows,cols)

output:- (145, 33)

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145 entries, 0 to 144
Data columns (total 33 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   STUDENTID       145 non-null    object
 1   AGE             145 non-null    int64
 2   GENDER          145 non-null    int64
 3   HS_TYPE         145 non-null    int64
 4   SCHOLARSHIP     145 non-null    int64
 5   WORK            145 non-null    int64
 6   ACTIVITY        145 non-null    int64
 7   PARTNER         145 non-null    int64
 8   SALARY          145 non-null    int64
 9   TRANSPORT       145 non-null    int64
 10  LIVING          145 non-null    int64
 11  MOTHER_EDU      145 non-null    int64
 12  FATHER_EDU      145 non-null    int64
 13  #_SIBLINGS      145 non-null    int64
 14  KIDS            145 non-null    int64
 15  MOTHER_JOB      145 non-null    int64
 16  FATHER_JOB      145 non-null    int64
 17  STUDY_HRS       145 non-null    int64
 18  READ_FREQ       145 non-null    int64
 19  READ_FREQ_SCI   145 non-null    int64
 20  ATTEND_DEPT     145 non-null    int64
 21  IMPACT          145 non-null    int64
 22  ATTEND          145 non-null    int64
 23  PREP_STUDY      145 non-null    int64
 24  PREP_EXAM       145 non-null    int64
 25  NOTES           145 non-null    int64
 26  LISTENS         145 non-null    int64
 27  LIKES_DISCUSS   145 non-null    int64
 28  CLASSROOM       145 non-null    int64
 29  CUML_GPA        145 non-null    int64
 30  EXP_GPA         145 non-null    int64
 31  COURSE ID       145 non-null    int64
 32  GRADE           145 non-null    int64
dtypes: int64(32), object(1)
memory usage: 37.5+ KB
```

df.describe().T.style.background_gradient(cmap = "Oranges")

	count	mean	std	min	25%	50%	75%	max
AGE	145.000000	1.620690	0.613154	1.000000	1.000000	2.000000	2.000000	3.000000
GENDER	145.000000	1.600000	0.491596	1.000000	1.000000	2.000000	2.000000	2.000000
HS_TYPE	145.000000	1.944828	0.537216	1.000000	2.000000	2.000000	2.000000	3.000000
SCHOLARSHIP	145.000000	3.572414	0.805750	1.000000	3.000000	3.000000	4.000000	5.000000
WORK	145.000000	1.662069	0.474644	1.000000	1.000000	2.000000	2.000000	2.000000
ACTIVITY	145.000000	1.600000	0.491596	1.000000	1.000000	2.000000	2.000000	2.000000
PARTNER	145.000000	1.579310	0.495381	1.000000	1.000000	2.000000	2.000000	2.000000
SALARY	145.000000	1.627586	1.020245	1.000000	1.000000	1.000000	2.000000	5.000000
TRANSPORT	145.000000	1.620690	1.061112	1.000000	1.000000	1.000000	2.000000	4.000000
LIVING	145.000000	1.731034	0.783999	1.000000	1.000000	2.000000	2.000000	4.000000
MOTHER_EDU	145.000000	2.282759	1.223062	1.000000	1.000000	2.000000	3.000000	6.000000
FATHER_EDU	145.000000	2.634483	1.147544	1.000000	2.000000	3.000000	3.000000	6.000000
#_SIBLINGS	145.000000	2.806897	1.360640	1.000000	2.000000	3.000000	4.000000	5.000000
KIDS	145.000000	1.172414	0.490816	1.000000	1.000000	1.000000	1.000000	3.000000
MOTHER_JOB	145.000000	2.358621	0.805156	1.000000	2.000000	2.000000	2.000000	5.000000
FATHER_JOB	145.000000	2.806897	1.329664	1.000000	2.000000	3.000000	4.000000	5.000000
STUDY_HRS	145.000000	2.200000	0.917424	1.000000	2.000000	2.000000	3.000000	5.000000
READ_FREQ	145.000000	1.944828	0.562476	1.000000	2.000000	2.000000	2.000000	3.000000
READ_FREQ_SCI	145.000000	2.013793	0.539884	1.000000	2.000000	2.000000	2.000000	3.000000
ATTEND_DEPT	145.000000	1.213793	0.411404	1.000000	1.000000	1.000000	1.000000	2.000000
IMPACT	145.000000	1.206897	0.588035	1.000000	1.000000	1.000000	1.000000	3.000000
ATTEND	145.000000	1.241379	0.429403	1.000000	1.000000	1.000000	1.000000	2.000000
PREP_STUDY	145.000000	1.337931	0.614870	1.000000	1.000000	1.000000	2.000000	3.000000
PREP_EXAM	145.000000	1.165517	0.408483	1.000000	1.000000	1.000000	1.000000	3.000000
NOTES	145.000000	2.544828	0.564940	1.000000	2.000000	3.000000	3.000000	3.000000
LISTENS	145.000000	2.055172	0.674736	1.000000	2.000000	2.000000	3.000000	3.000000
LIKES_DISCUSS	145.000000	2.393103	0.604343	1.000000	2.000000	2.000000	3.000000	3.000000
CLASSROOM	145.000000	1.806897	0.810492	1.000000	1.000000	2.000000	2.000000	3.000000
CUML_GPA	145.000000	3.124138	1.301083	1.000000	2.000000	3.000000	4.000000	5.000000
EXP_GPA	145.000000	2.724138	0.916536	1.000000	2.000000	3.000000	3.000000	4.000000
COURSE ID	145.000000	4.131034	3.260145	1.000000	1.000000	3.000000	7.000000	9.000000
GRADE	145.000000	3.227586	2.197678	0.000000	1.000000	3.000000	5.000000	7.000000

df["COURSE ID"].unique()

output:- array([1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=int64) df.describe(include=object)

Out[11]:

	STUDENTID
count	145
unique	145
top	STUDENT1
freq	1

df = df.drop('STUDENTID', axis=1)

#checking the duplicate
duplicate = df[df.duplicated()]
print("Duplicate Rows :")

duplicate

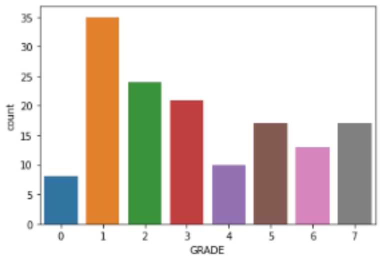
```
Duplicate Rows :
```

	AGE	GENDER	HS_TYPE	SCHOLARSHIP	WORK	ACTIVITY	PARTNER	SALARY	TRANSPORT	LIVING	...	PREP_STUDY	PREP_EXAM	NOTES	LISTENS	I

0 rows × 32 columns

sns.countplot(df['GRADE'],label="Count")
plt.show()

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```



X = df.drop('GRADE', axis=1)
y = df['GRADE']

# list discrete features that have integer dtypes for using MI (Mutual Information)
discrete_features = X.dtypes == int

def make_mi_scores(X, y, discrete_features):

```
    mi_scores = mutual_info_classif(X, y, discrete_features=discrete_features)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores

mi_scores = make_mi_scores(X, y, discrete_features)
mi_scores  # show a few features with their MI scores
```

```
Out[16]:  COURSE ID          0.491416
          GENDER             0.176673
          MOTHER_EDU         0.157819
          EXP_GPA            0.152561
          CUML_GPA           0.144045
          ATTEND             0.134883
          AGE                0.132892
          FATHER_JOB         0.082216
          SCHOLARSHIP        0.073717
          IMPACT             0.073269
          TRANSPORT          0.058359
          LIVING             0.037292
          PARTNER            0.029098
          LIKES_DISCUSS      0.026543
          LISTENS            0.017192
          STUDY_HRS          0.008637
          WORK               0.006424
          ATTEND_DEPT        0.001855
          NOTES              0.001841
          READ_FREQ_SCI      0.000000
          READ_FREQ          0.000000
          ACTIVITY           0.000000
          MOTHER_JOB         0.000000
          PREP_STUDY         0.000000
          PREP_EXAM          0.000000
          KIDS               0.000000
          #_SIBLINGS         0.000000
          CLASSROOM          0.000000
          FATHER_EDU         0.000000
          HS_TYPE            0.000000
          SALARY             0.000000
          Name: MI Scores, dtype: float64
```

```
def drop_uninformative(df, mi_scores):
    return df.loc[:, mi_scores > 0]


X = drop_uninformative(X, mi_scores)
```

#k -means is **a centroid-based clustering algorithm, where we calculate the distance between each data point and a centroid to assign it to a cluster**. The goal is to identify the K number of groups in the dataset.

```
kmeans = KMeans(n_clusters=8, random_state=0)
X["Cluster"] = kmeans.fit_predict(X)
decision_tree = dtc(random_state=0)
```

decision_tree.fit(X,y)

decision tree is **a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks**. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

predict = cross_val_predict(estimator = decision_tree, X = X, y = y, cv = 5)

print("Classification Report: \n",classification_report(y, predict))

```
Classification Report:
              precision    recall  f1-score   support

           0       0.11      0.12      0.12         8
           1       0.31      0.37      0.34        35
           2       0.38      0.42      0.40        24
           3       0.29      0.33      0.31        21
           4       0.00      0.00      0.00        10
           5       0.13      0.12      0.12        17
           6       0.20      0.08      0.11        13
           7       0.44      0.47      0.46        17

    accuracy                           0.29       145
   macro avg       0.23      0.24      0.23       145
weighted avg       0.27      0.29      0.28       145
```

Random Forest:

#It combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

random_forest = RandomForestClassifier(random_state = 0)

random_forest.fit(X, y)

predict = cross_val_predict(estimator = random_forest, X = X, y = y, cv = 5)

print("Classification Report: \n",classification_report(y, predict))

```
Classification Report:
               precision    recall  f1-score   support

           0       0.33      0.25      0.29         8
           1       0.29      0.51      0.37        35
           2       0.36      0.33      0.35        24
           3       0.31      0.24      0.27        21
           4       0.00      0.00      0.00        10
           5       0.25      0.12      0.16        17
           6       0.29      0.15      0.20        13
           7       0.36      0.47      0.41        17

    accuracy                           0.31       145
   macro avg       0.27      0.26      0.26       145
weighted avg       0.29      0.31      0.29       145
```

KNN :

#k-NN, is a non-parametric, supervised learning classifier, which **uses proximity to make classifications or predictions about the grouping of an individual data point**.

knn = KNeighborsClassifier()

knn.fit(X,y)

predict = cross_val_predict(estimator = knn, X = X, y = y, cv = 5)

print("Classification Report: \n",classification_report(y, predict))

```
Classification Report:
               precision    recall  f1-score   support

           0       0.20      0.25      0.22         8
           1       0.32      0.60      0.42        35
           2       0.23      0.12      0.16        24
           3       0.22      0.10      0.13        21
           4       0.00      0.00      0.00        10
           5       0.27      0.18      0.21        17
           6       0.18      0.23      0.20        13
           7       0.24      0.24      0.24        17

    accuracy                           0.26       145
   macro avg       0.21      0.21      0.20       145
weighted avg       0.23      0.26      0.23       145
```

Hyperparameter Tuning :-

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.metrics import accuracy_score, confusion_matrix

 # load the dataset

df = pd.read_csv('student_prediction.csv')

 # extract features and labels

X = df.iloc[:, 1:-1]

y = df.iloc[:, -1]

create the classifier

clf = DecisionTreeClassifier()

 # define the hyperparameters to tune

parameters = {

  'criterion': ['gini', 'entropy'],

  'max_depth': [2, 4, 6, 8],

  'min_samples_split': [2, 4, 6, 8],

  'min_samples_leaf': [1, 2, 3, 4]

}

 # create the GridSearchCV object

grid_search = GridSearchCV(clf, param_grid=parameters, cv=5, n_jobs=-1)

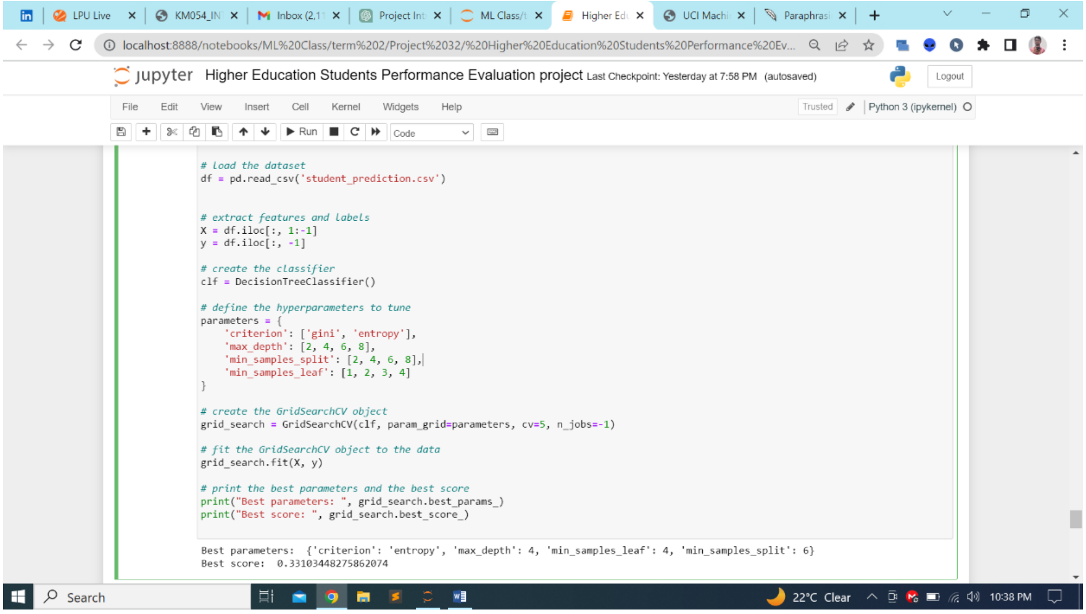
 # fit the GridSearchCV object to the data

grid_search.fit(X, y)

 # print the best parameters and the best score

print("Best parameters: ", grid_search.best_params_)

print("Best score: ", grid_search.best_score_)



GaussianNB :

#Gaussian Naive Bayes **assumes that each parameter (also called features or predictors) has an independent capacity of predicting the output variable**. The combination of the prediction for all parameters is the final prediction, that returns a probability of the dependent variable to be classified in each group.

gnb = GaussianNB()

gnb.fit(X,y)

predict = cross_val_predict(estimator = gnb, X = X, y = y, cv = 5)

print("Classification Report: \n",classification_report(y, predict))

Classification Report:

	precision	recall	f1-score	support
0	0.14	0.50	0.22	8
1	0.38	0.17	0.24	35
2	0.33	0.08	0.13	24
3	0.14	0.05	0.07	21
4	0.00	0.00	0.00	10
5	0.10	0.06	0.07	17
6	0.17	0.31	0.22	13
7	0.24	0.71	0.36	17
accuracy			0.21	145
macro avg	0.19	0.23	0.16	145
weighted avg	0.23	0.21	0.17	145

SVC

#SVC, or Support Vector Classifier, is **a supervised machine learning algorithm typically used for classification tasks**. SVC works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides the data into two classes.

```
scv = SVC()
scv.fit(X,y)
predict = cross_val_predict(estimator = scv, X = X, y = y, cv = 5)
print("Classification Report: \n",classification_report(y, predict))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.33      0.12      0.18         8
           1       0.30      0.77      0.43        35
           2       0.18      0.08      0.11        24
           3       0.33      0.05      0.08        21
           4       0.00      0.00      0.00        10
           5       0.17      0.06      0.09        17
           6       0.00      0.00      0.00        13
           7       0.33      0.59      0.43        17

    accuracy                           0.29       145
   macro avg       0.21      0.21      0.17       145
weighted avg       0.23      0.29      0.20       145

C:\Users\Dell\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Dell\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Conclusion:-

Overall, the Higher Education Students Performance Assessment Dataset is a useful tool for academics and teachers who want to comprehend and enhance higher education student achievement. Any inferences made from this dataset, however, would rely on the precise study question posed and the techniques employed to evaluate the data. To ensure the validity of their conclusions, researchers must carefully analyse the constraints and potential biases of the data and apply the proper statistical techniques.

Review Paper On This Project : - [Review Paper Link]

Thank you