BRAC UNIVERSITY

Inspiring Excellence

Name	**:**	Afia Anjum, Md. Tasnim Muttaki
Id	**:**	21301078, 21101216
Subject	**:**	Artificial Intelligence
Course code	**:**	CSE 422
Section:	10	

Table of Contents

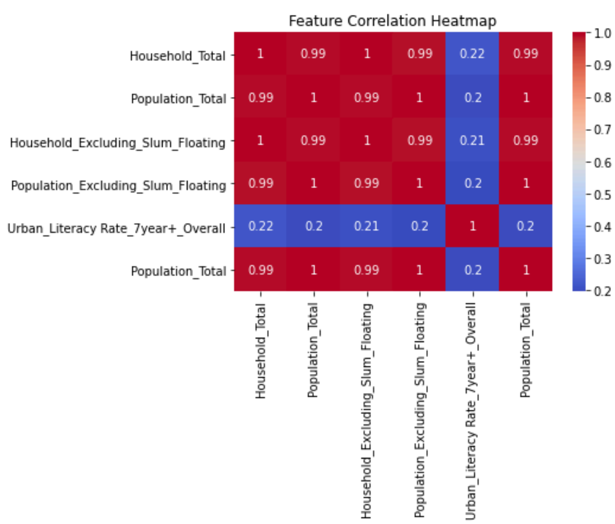
Introduction:

This project analyzes and models data from the Bangladesh Census dataset to solve classification and regression problems. Using household and demographic data, it shows urban literacy rates and the total population. The study contributes to the knowledge of socioeconomic patterns and the application of machine learning techniques to make reliable predictions.

Dataset Description:

- Source link/reference link: https://data.humdata.org/dataset/bangladesh-subnational-boundaries-and-tabular-data
- **Dataset Description:**
    - **Number of features: 4** (`Household_Total`, `Population_Total`, `Household_Excluding_Slum_Floating`, `Population_Excluding_Slum_Floating`)
    - **Problem Type:**
        - Classification
        - Regression
    - **Number of data points:** 64,445 (row, column)
    - **Feature Type:** Quantitive
- **Correlation analysis:**

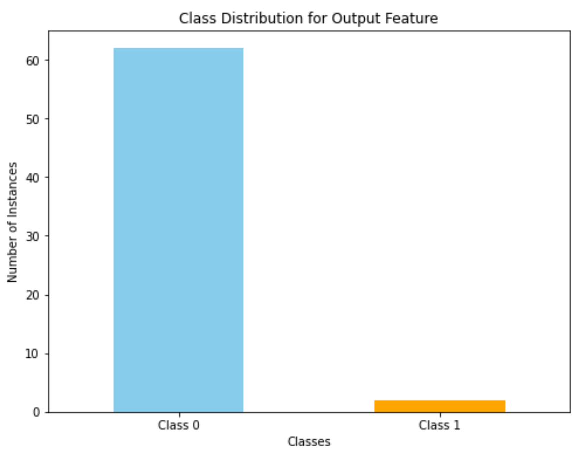
In [75]:
```python
import seaborn as sns
correlation_matrix = selected_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()
```



- **Imbalance dataset:**

```
In [4]: import pandas as pd
import matplotlib.pyplot as plt
data_path =r'C:\Users\Muttaki\Downloads\bangladesh_bbs_population-and-housing-census-dataset_2022_admin-02.xlsx'
data = pd.read_excel(data_path)
classification_column = 'Urban_Literacy Rate_7year+_Overall'
classification_labels = (data[classification_column] > 70).astype(int)
print("Imbalanced Dataset:")
class_counts = classification_labels.value_counts()
print("Class Distribution:")
print(class_counts)
plt.figure(figsize=(8, 6))
class_counts.plot(kind='bar', color=['skyblue', 'orange'])
plt.title("Class Distribution for Output Feature")
plt.xlabel("Classes")
plt.ylabel("Number of Instances")
plt.xticks([0, 1], ['Class 0', 'Class 1'], rotation=0)
plt.show()
```

```
Imbalanced Dataset:
Class Distribution:
1    62
0     2
Name: Urban_Literacy Rate_7year+_Overall, dtype: int64
```



DAta pre-prossessing:

- **Faults:**
    - Missing Values: There is no missing values or null values
    - Data Types: Here eatures and labeles are quantitive thats why int need scaling

- **Solution:**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.metrics import accuracy_score, mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
data_path = r'C:\Users\Muttaki\Downloads\bangladesh_bbs_population-and-housing-census-dataset_2022_admin-02.xlsx'
data = pd.read_excel(data_path)
selected_data = selected_data.dropna()
selected_data = selected_data.dropna()
print("=====Dataset preview=====")
print(data.head())
selected_data = data[feature_columns + [classification_column, regression_label_column]] #filtering
print("=====Selected Data===== ")
print(selected_data)
print("===Miss_values===")
print(selected_data.isnull().sum())
```

```
=====Dataset preview=====
   Division   District  Division_Geocode  District_Geocode  Household_Total  \
0  Barishal    Barguna                10                 4           255390
1    Khulna   Bagerhat                40                 1           408840
2  Barishal   Barishal                10                 6           629626
3  Barishal      Bhola                10                 9           449057
4  Barishal  Jhalokati                10                42           162401

   Population_Total  Household_Excluding_Slum_Floating  \
0           1010531                             254895
1           1613076                             406058
2           2570446                             620735
3           1932518                             448944
4            661160                             162212

   Population_Excluding_Slum_Floating  Household_Slum  Population_Slum  ...  \
0                             1008695             451             1789  ...
1                             1603126            2605             9763  ...
2                             2537292            8160            31338  ...
3                             1932138              74              339  ...
4                              660520             143              588  ...

   Number of Person & Avg HH Size_# of HH with 3-Person  \
0                                              57097
1                                              96057
```

2	135271
3	89014
4	34540

	Number of Person & Avg HH Size_# of HH with 4-Person \
0	68828
1	109638
2	165285
3	118802
4	41357

	Number of Person & Avg HH Size_# of HH with 5-Person \
0	43330
1	68442
2	110132
3	92260
4	27852

	Number of Person & Avg HH Size_# of HH with 6-Person \
0	20901
1	32081
2	54039
3	46329
4	14052

	Number of Person & Avg HH Size_# of HH with 7-Person \
0	8395
1	12933
2	23613
3	20396
4	6339

	Number of Person & Avg HH Size_# of HH with 8-Person \
0	3499
1	5722
2	10535
3	9347
4	3084

	Number of Person & Avg HH Size_# of HH with 9-Person \
0	1583
1	2560
2	5289
3	4588
4	1529

```
   Number of Person & Avg HH Size_# of HH with 10-Person  \
0                                               772
1                                              1324
2                                              2819
3                                              2375
4                                               815

   Number of Person & Avg HH Size_# of HH with 10-Person +  \
0                                               767
1                                              1236
2                                              2981
3                                              2574
4                                               904

   Number of Person & Avg HH Size_Average Household Size
0                                              3.92
1                                              3.89
2                                              4.02
3                                              4.27
4                                              4.02

[5 rows x 445 columns]
=====Selected Data=====
    Household_Total  Population_Total  Household_Excluding_Slum_Floating  \
0            255390           1010531                             254895
1            408840           1613076                             406058
2            629626           2570446                             620735
3            449057           1932518                             448944
4            162401            661160                             162212
..              ...               ...                                ...
59           281627           1179843                             280851
60           834307           3169614                             826692
61           382400           1533895                             381262
62           528550           2695496                             527545
63           746854           3857123                             722553

    Population_Excluding_Slum_Floating  Urban_Literacy Rate_7year+_Overall  \
0                              1008695                               85.21
1                              1603126                               82.23
2                              2537292                               85.43
3                              1932138                               72.58
4                               660520                               86.38
..                                 ...                                 ...
59                             1176974                               79.00
60                             3141432                               78.19
61                             1529809                               80.36
```

```
62                2691484                72.89
63                3753617                81.47

   Population_Total
0         1010531
1         1613076
2         2570446
3         1932518
4          661160
..            ...
59        1179843
60        3169614
61        1533895
62        2695496
63        3857123

[64 rows x 6 columns]
===Miss_values===
Household_Total                      0
Population_Total                     0
Household_Excluding_Slum_Floating    0
Population_Excluding_Slum_Floating   0
Urban_Literacy Rate_7year+_Overall   0
Population_Total                     0
dtype: int64
```

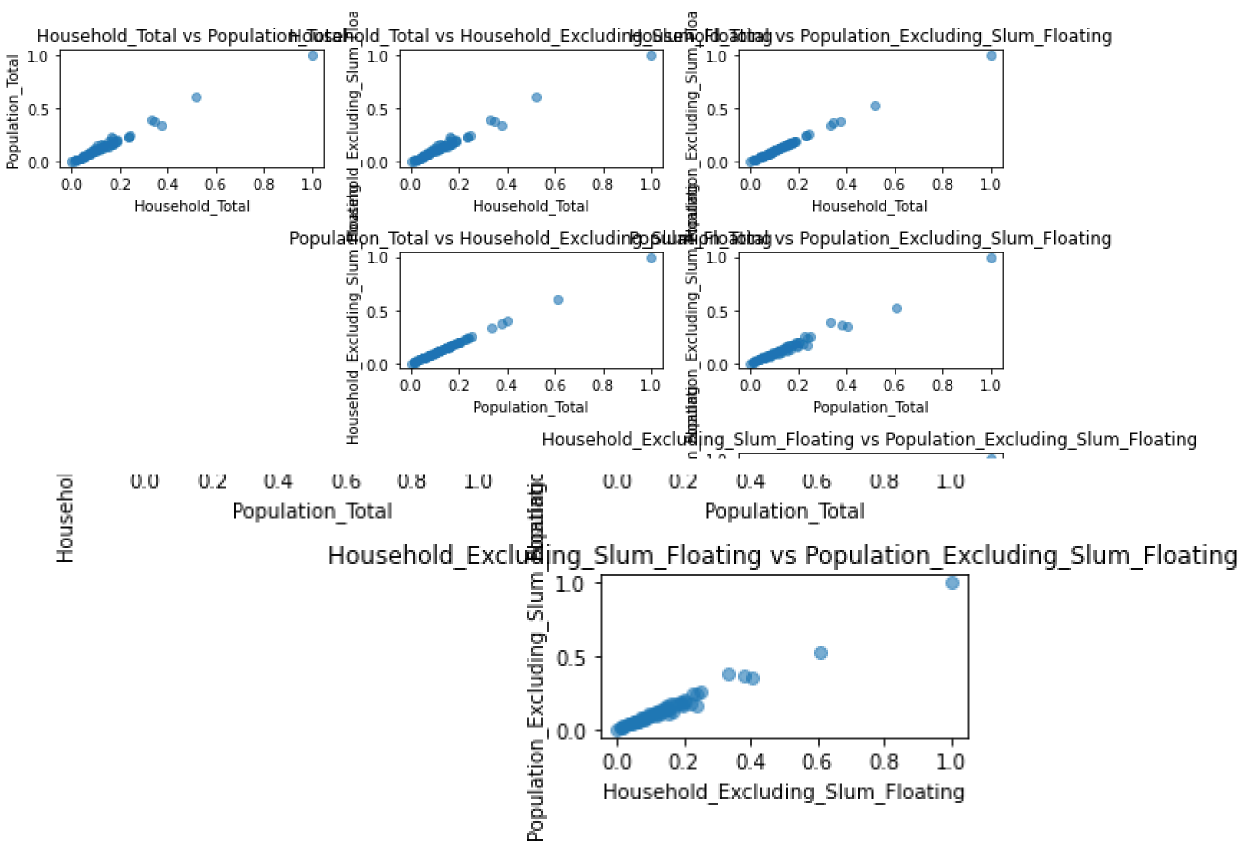
Feature scaling:

```
feature_columns = [
    'Household_Total',
    'Population_Total',
    'Household_Excluding_Slum_Floating',
    'Population_Excluding_Slum_Floating']
classification_column = 'Urban_Literacy Rate_7year+_Overall'    #lr
regression_label_column = 'Population_Total'    #lr
features = selected_data[feature_columns]
classification_labels = (selected_data[classification_column] > 70).astype(int)  # 0.1classify
regression_labels = selected_data[regression_label_column]
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(features) # f_sc
print("--------feature_columns------")
print(feature_columns)
print("------features-------")
print(features)
print("-----classification_column-------")
print(classification_column)
print("------regression_label_column-------")
print(regression_label_column)
print("-------classification-labels---------")
print(classification_labels)
print("------regression_labels------")
print(regression_labels)
print("--------scaler-------")
print(scaler)
print("---------scaled_features-----")
print(scaled_features)
```

```
[[0.03798212 0.03714326 0.03714326 0.04004783 0.03891532]
 [0.077037   0.07941646 0.07941646 0.08072344 0.08275346]
 [0.13322971 0.14658337 0.14658337 0.13848967 0.15164642]
 [0.08727271 0.10182778 0.10182778 0.09226339 0.10701747]
 [0.01431529 0.01263218 0.01263218 0.01510828 0.01323808]
 [0.         0.         0.         0.         0.        ]
 [0.08108451 0.08742693 0.08742693 0.0856162  0.09177853]
 [0.04895159 0.05030934 0.05030934 0.05175274 0.05288129]
 [0.154342   0.19822768 0.19822768 0.16256923 0.20772963]
 [0.13470716 0.15116481 0.15116481 0.14156854 0.15796818]
 [0.23390809 0.22823653 0.22823653 0.24677166 0.239434  ]
 [0.51863309 0.60955562 0.60955562 0.52589964 0.61832696]
 [0.12240989 0.16432079 0.16432079 0.12857122 0.17181593]
 [0.33117448 0.40208172 0.40208172 0.34848826 0.42082656]
 [0.06897507 0.08192951 0.08192951 0.07103048 0.08423846]
 [0.01612869 0.01634767 0.01634767 0.01699941 0.01712504]
 [0.08989088 0.10220874 0.10220874 0.09503281 0.10741947]
 [0.05613494 0.05282513 0.05282513 0.05901021 0.05519085]
 [0.17050149 0.2205995  0.2205995  0.17899441 0.23039698]
 [0.0120453  0.01167986 0.01167986 0.0123795  0.01186791]
 [1.         1.         1.         1.         1.        ]
 [0.10682434 0.11798939 0.11798939 0.11241243 0.12353173]
 [0.37505567 0.33551844 0.33551844 0.38542843 0.3441169 ]
 [0.0515527  0.05710496 0.05710496 0.0544678  0.05998234]
 [0.16665377 0.19549594 0.19549594 0.17609825 0.20544115]
 [0.05271404 0.05696254 0.05696254 0.05568702 0.05982164]
 [0.18600306 0.19883615 0.19883615 0.19440972 0.20671692]
 [0.07313889 0.0755542  0.0755542  0.07728858 0.07939748]
 [0.0746932  0.0802822  0.0802822  0.07881133 0.08424377]
 [0.04167636 0.03334773 0.03334773 0.04363849 0.03463866]
 [0.2333927  0.24050297 0.24050297 0.24433591 0.25051236]
 [0.13116435 0.147566   0.147566   0.13803008 0.15443852]
 [0.04811832 0.04972163 0.04972163 0.05042211 0.05182543]
 [0.05161735 0.05707023 0.05707023 0.05441936 0.05981331]
 [0.15121405 0.14600717 0.14600717 0.15945969 0.15306268]
 [0.24320949 0.24951614 0.24951614 0.2566519  0.26179007]
 [0.17609108 0.182062   0.182062   0.18424721 0.18951215]
 [0.10514914 0.10697252 0.10697252 0.1110097  0.11229282]
 [0.0981727  0.1317408  0.1317408  0.10364163 0.13831136]
 [0.14372452 0.14959587 0.14959587 0.14691201 0.15233803]
 [0.11686789 0.11706422 0.11706422 0.12290969 0.12244859]
 [0.13893613 0.14162266 0.14162266 0.14670267 0.1486653 ]
 [0.03766754 0.0387277  0.0387277  0.03979086 0.04065776]
 [0.02269408 0.01573287 0.01573287 0.02397946 0.01650239]
 [0.02278011 0.02157807 0.02157807 0.02400959 0.02259147]
 [0.11722752 0.12035392 0.12035392 0.12348203 0.12606749]
```

```
[0.34480004 0.38010754 0.38010754 0.36267223 0.3976218 ]
[0.16780035 0.16160786 0.16160786 0.17734977 0.16981006]
[0.11256766 0.12935312 0.12935312 0.11881906 0.1357523 ]
[0.07380699 0.0716133  0.0716133  0.07801887 0.07526258]
[0.10073641 0.09673461 0.09673461 0.10647966 0.10166579]
[0.08701082 0.09502319 0.09502319 0.09120051 0.09902096]
[0.12714586 0.12965529 0.12965529 0.13412539 0.13598063]
[0.16222679 0.17037933 0.17037933 0.17063154 0.17821623]
[0.17029151 0.17075713 0.17075713 0.17856468 0.17812759]
[0.06003254 0.06646043 0.06646043 0.06275075 0.06909879]
[0.18735986 0.20181575 0.20181575 0.19774529 0.21177205]
[0.08658528 0.11522293 0.11522293 0.09092174 0.12042745]
[0.10166487 0.11305653 0.11305653 0.10746451 0.11882454]
[0.04465975 0.04902181 0.04902181 0.04703219 0.05132557]
[0.18532351 0.18861964 0.18861964 0.19390949 0.19620059]
[0.0703077  0.0738613  0.0738613  0.07405122 0.07734648]
[0.10750465 0.1553566  0.1553566  0.1134137  0.16301778]
[0.16306566 0.23685372 0.23685372 0.16588731 0.24134806]]
```

S



Dataset Spliting:

```
feature_columns = [
    'Household_Total',
    'Population_Total',
    'Household_Excluding_Slum_Floating',
    'Population_Excluding_Slum_Floating']
classification_column = 'Urban_Literacy Rate_7year+_Overall'   #lr
regression_label_column = 'Population_Total'   #lr
features = selected_data[feature_columns]
classification_labels = (selected_data[classification_column] > 70).astype(int)  # 0.1classify
regression_labels = selected_data[regression_label_column]
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(features) # f_sc
print('--------feature_columns------')
print(feature_columns)
print("------features-------")
print(features)
print("-----classification_column-------")
print(classification_column)
print("------regression_label_column-------")
print(regression_label_column)
print("-------classification-labels---------")
print(classification_labels)
print('------regression_labels------')
print(regression_labels)
print("--------scaler-------")
print(scaler)
print("---------scaled_features-----")
print(scaled_features)
```

```
Population_Total
-------classification-labels---------
0     1
1     1
2     1
3     1
4     1
     ..
59    1
60    1
61    1
62    1
63    1
Name: Urban_Literacy Rate_7year+_Overall, Length: 64, dtype: int32
```

In [63]:
```python
#data split data train_taste
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(
    scaled_features, classification_labels, test_size=0.25, random_state=0)
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    scaled_features, regression_labels, test_size=0.25, random_state=0)
print("-----X_train_clf, X_test_clf, y_train_clf, y_test_clf-------")
print(X_train_clf, X_test_clf, y_train_clf, y_test_clf)
print("------X_train_reg, X_test_reg, y_train_reg, y_test_reg -------")
print(X_train_reg, X_test_reg, y_train_reg, y_test_reg )
```

```
----X_train_clf, X_test_clf, y_train_clf, y_test_clf-------
[[0.23390809 0.22823653 0.22823653 0.24677166 0.239434  ]
 [0.24320949 0.24951614 0.24951614 0.2566519  0.26179007]
 [0.12714586 0.12965529 0.12965529 0.13412539 0.13598063]
 [0.34480004 0.38010754 0.38010754 0.36267223 0.3976218 ]
 [0.2333927  0.24050297 0.24050297 0.24433591 0.25051236]
 [0.04895159 0.05030934 0.05030934 0.05175274 0.05288129]
 [0.06897507 0.08192951 0.08192951 0.07103048 0.08423846]
 [0.07313889 0.0755542  0.0755542  0.07728858 0.07939748]
 [0.16306566 0.23685372 0.23685372 0.16588731 0.24134806]
 [0.06003254 0.06646043 0.06646043 0.06275075 0.06909879]
 [0.13893613 0.14162266 0.14162266 0.14670267 0.1486653 ]
 [0.03766754 0.0387277  0.0387277  0.03979086 0.04065776]
 [0.10166487 0.11305653 0.11305653 0.10746451 0.11882454]
 [0.17050149 0.2205995  0.2205995  0.17899441 0.23039698]
 [0.18532351 0.18861964 0.18861964 0.19390949 0.19620059]
 [0.04811832 0.04972163 0.04972163 0.05042211 0.05182543]
 [0.01612869 0.01634767 0.01634767 0.01699941 0.01712504]
 [0.         0.         0.         0.         0.        ]
 [0.08989088 0.10220874 0.10220874 0.09503281 0.10741947]
 [1.         1.         1.         1.         1.        ]
 [0.18735986 0.20181575 0.20181575 0.19774529 0.21177205]
 [0.154342   0.19822768 0.19822768 0.16256923 0.20772963]
 [0.33117448 0.40208172 0.40208172 0.34848826 0.42082656]
 [0.05271404 0.05696254 0.05696254 0.05568702 0.05982164]
 [0.10514914 0.10697252 0.10697252 0.1110097  0.11229282]
 [0.05613494 0.05282513 0.05282513 0.05901021 0.05519085]
 [0.11256766 0.12935312 0.12935312 0.11881906 0.1357523 ]
 [0.08701082 0.09502319 0.09502319 0.09120051 0.09902096]
 [0.08658528 0.11522293 0.11522293 0.09092174 0.12042745]
 [0.0981727  0.1317408  0.1317408  0.10364163 0.13831136]
 [0.077037   0.07941646 0.07941646 0.08072344 0.08275346]
 [0.12240989 0.16432079 0.16432079 0.12857122 0.17181593]
 [0.07380699 0.0716133  0.0716133  0.07801887 0.07526258]
 [0.16665377 0.19549594 0.19549594 0.17609825 0.20544115]
 [0.08108451 0.08742693 0.08742693 0.0856162  0.09177853]
```

[0.0515527 0.05710496 0.05710496 0.0544678 0.05998234]
[0.17609108 0.182062   0.182062   0.18424721 0.18951215]
[0.10073641 0.09673461 0.09673461 0.10647966 0.10166579]
[0.10682434 0.11798939 0.11798939 0.11241243 0.12353173]
[0.0120453 0.01167986 0.01167986 0.0123795 0.01186791]
[0.13470716 0.15116481 0.15116481 0.14156854 0.15796818]
[0.14372452 0.14959587 0.14959587 0.14691201 0.15233803]
[0.04465975 0.04902181 0.04902181 0.04703219 0.05132557]
[0.08727271 0.10182778 0.10182778 0.09226339 0.10701747]
[0.03798212 0.03714326 0.03714326 0.04004783 0.03891532]
[0.16222679 0.17037933 0.17037933 0.17063154 0.17821623]
[0.16780035 0.16160786 0.16160786 0.17734977 0.16981006]
[0.02278011 0.02157807 0.02157807 0.02400959 0.02259147]] [[0.11722752 0.12035392
0.12035392 0.12348203 0.12606749]
[0.04167636 0.03334773 0.03334773 0.04363849 0.03463866]
[0.02269408 0.01573287 0.01573287 0.02397946 0.01650239]
[0.0703077 0.0738613 0.0738613 0.07405122 0.07734648]
[0.15121405 0.14600717 0.14600717 0.15945969 0.15306268]
[0.05161735 0.05707023 0.05707023 0.05441936 0.05981331]
[0.13116435 0.147566   0.147566   0.13803008 0.15443852]
[0.11686789 0.11706422 0.11706422 0.12290969 0.12244859]
[0.18600306 0.19883615 0.19883615 0.19440972 0.20671692]
[0.10750465 0.1553566 0.1553566 0.1134137 0.16301778]
[0.37505567 0.33551844 0.33551844 0.38542843 0.3441169]
[0.13322971 0.14658337 0.14658337 0.13848967 0.15164642]
[0.51863309 0.60955562 0.60955562 0.52589964 0.61832696]
[0.0746932 0.0802822 0.0802822 0.07881133 0.08424377]
[0.17029151 0.17075713 0.17075713 0.17856468 0.17812759]
[0.01431529 0.01263218 0.01263218 0.01510828 0.01323808]] 10   1
35   1
52   1
46   1
30   1
7   1
14   1
27   1
63   1
55   1
41   0
42   1
58   1
18   1
60   1
32   1

15   1
5   1
16   1
20   1
56   1
8   1
13   1
25   1
37   1
17   1
48   1
51   1
57   1
38   1
1   1
12   1
49   0
24   1
6   1
23   1
36   1
50   1
21   1
19   1
9   1
39   1
59   1
3   1
0   1
53   1
47   1
44   1
Name: Urban_Literacy Rate_7year+_Overall, dtype: int32 45   1
29   1
43   1
61   1
34   1
33   1
31   1
40   1
26   1
62   1
22   1
2   1

```
11    1
28    1
54    1
4     1
Name: Urban_Literacy Rate_7year+_Overall, dtype: int32
------X_train_reg, X_test_reg, y_train_reg, y_test_reg -------
[[0.23390809 0.22823653 0.22823653 0.24677166 0.239434  ]
 [0.24320949 0.24951614 0.24951614 0.2566519  0.26179007]
 [0.12714586 0.12965529 0.12965529 0.13412539 0.13598063]
 [0.34480004 0.38010754 0.38010754 0.36267223 0.3976218 ]
 [0.2333927  0.24050297 0.24050297 0.24433591 0.25051236]
 [0.04895159 0.05030934 0.05030934 0.05175274 0.05288129]
 [0.06897507 0.08192951 0.08192951 0.07103048 0.08423846]
 [0.07313889 0.0755542  0.0755542  0.07728858 0.07939748]
 [0.16306566 0.23685372 0.23685372 0.16588731 0.24134806]
 [0.06003254 0.06646043 0.06646043 0.06275075 0.06909879]
 [0.13893613 0.14162266 0.14162266 0.14670267 0.1486653 ]
 [0.03766754 0.0387277  0.0387277  0.03979086 0.04065776]
 [0.10166487 0.11305653 0.11305653 0.10746451 0.11882454]
 [0.17050149 0.2205995  0.2205995  0.17899441 0.23039698]
 [0.18532351 0.18861964 0.18861964 0.19390949 0.19620059]
 [0.04811832 0.04972163 0.04972163 0.05042211 0.05182543]
 [0.01612869 0.01634767 0.01634767 0.01699941 0.01712504]
 [0.         0.         0.         0.         0.        ]
 [0.08989088 0.10220874 0.10220874 0.09503281 0.10741947]
 [1.         1.         1.         1.         1.        ]
 [0.18735986 0.20181575 0.20181575 0.19774529 0.21177205]
 [0.154342   0.19822768 0.19822768 0.16256923 0.20772963]
 [0.33117448 0.40208172 0.40208172 0.34848826 0.42082656]
 [0.05271404 0.05696254 0.05696254 0.05568702 0.05982164]
 [0.10514914 0.10697252 0.10697252 0.1110097  0.11229282]
 [0.05613494 0.05282513 0.05282513 0.05901021 0.05519085]
 [0.11256766 0.12935312 0.12935312 0.11881906 0.1357523 ]
 [0.08701082 0.09502319 0.09502319 0.09120051 0.09902096]
 [0.08658528 0.11522293 0.11522293 0.09092174 0.12042745]
 [0.0981727  0.1317408  0.1317408  0.10364163 0.13831136]
 [0.077037   0.07941646 0.07941646 0.08072344 0.08275346]
 [0.12240989 0.16432079 0.16432079 0.12857122 0.17181593]
 [0.07380699 0.0716133  0.0716133  0.07801887 0.07526258]
 [0.16665377 0.19549594 0.19549594 0.17609825 0.20544115]
 [0.08108451 0.08742693 0.08742693 0.0856162  0.09177853]
 [0.0515527  0.05710496 0.05710496 0.0544678  0.05998234]
 [0.17609108 0.182062   0.182062   0.18424721 0.18951215]
 [0.10073641 0.09673461 0.09673461 0.10647966 0.10166579]
```

```
 [0.10682434 0.11798939 0.11798939 0.11241243 0.12353173]
 [0.0120453  0.01167986 0.01167986 0.0123795  0.01186791]
 [0.13470716 0.15116481 0.15116481 0.14156854 0.15796818]
 [0.14372452 0.14959587 0.14959587 0.14691201 0.15233803]
 [0.04465975 0.04902181 0.04902181 0.04703219 0.05132557]
 [0.08727271 0.10182778 0.10182778 0.09226339 0.10701747]
 [0.03798212 0.03714326 0.03714326 0.04004783 0.03891532]
 [0.16222679 0.17037933 0.17037933 0.17063154 0.17821623]
 [0.16780035 0.16160786 0.16160786 0.17734977 0.16981006]
 [0.02278011 0.02157807 0.02157807 0.02400959 0.02259147]] [[0.11722752 0.12035392
0.12035392 0.12348203 0.12606749]
 [0.04167636 0.03334773 0.03334773 0.04363849 0.03463866]
 [0.02269408 0.01573287 0.01573287 0.02397946 0.01650239]
 [0.0703077  0.0738613  0.0738613  0.07405122 0.07734648]
 [0.15121405 0.14600717 0.14600717 0.15945969 0.15306268]
 [0.05161735 0.05707023 0.05707023 0.05441936 0.05981331]
 [0.13116435 0.147566   0.147566   0.13803008 0.15443852]
 [0.11686789 0.11706422 0.11706422 0.12290969 0.12244859]
 [0.18600306 0.19883615 0.19883615 0.19440972 0.20671692]
 [0.10750465 0.1553566  0.1553566  0.1134137  0.16301778]
 [0.37505567 0.33551844 0.33551844 0.38542843 0.3441169 ]
 [0.13322971 0.14658337 0.14658337 0.13848967 0.15164642]
 [0.51863309 0.60955562 0.60955562 0.52589964 0.61832696]
 [0.0746932  0.0802822  0.0802822  0.07881133 0.08424377]
 [0.17029151 0.17075713 0.17075713 0.17856468 0.17812759]
 [0.01431529 0.01263218 0.01263218 0.01510828 0.01323808]]     Population_Total
Population_Total
10     3734297     3734297
35     4037608     4037608
52     2329160     2329160
46     5899005     5899005
30     3909138     3909138
7      1198195     1198195
14     1648896     1648896
27     1558025     1558025
63     3857123     3857123
55     1428406     1428406
41     2499738     2499738
42     1033115     1033115
58     2092568     2092568
18     3625442     3625442
60     3169614     3169614
32     1189818     1189818
15      714119      714119
```

5	481106	481106		
16	1937948	1937948		
20	14734701	14734701		
56	3357706	3357706		
8	3306563	3306563		
13	6212216	6212216		
25	1293027	1293027		
37	2005849	2005849		
17	1234054	1234054		
48	2324853	2324853		
51	1835528	1835528		
57	2123447	2123447		
38	2358886	2358886		
1	1613076	1613076		
12	2823268	2823268		
49	1501853	1501853		
24	3267626	3267626		
6	1727254	1727254		
23	1295057	1295057		
36	3076144	3076144		
50	1859922	1859922		
21	2162879	2162879		
19	647586	647586		
9	2635748	2635748		
39	2613385	2613385		
59	1179843	1179843		
3	1932518	1932518		
0	1010531	1010531		
53	2909624	2909624		
47	2784599	2784599		
44	788671	788671	Population_Total	Population_Total
45	2196582	2196582		
29	956431	956431		
43	705356	705356		
61	1533895	1533895		
34	2562233	2562233		
33	1294562	1294562		
31	2584452	2584452		
40	2149692	2149692		
26	3315236	3315236		
62	2695496	2695496		
22	5263450	5263450		
2	2570446	2570446		
11	9169465	9169465		

28	1625416	1625416
54	2915009	2915009
4	661160	661160

Model Train and Testing:

- **KNN:**

In [64]:
```
#KNN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_clf, y_train_clf)
knn_predictions = knn.predict(X_test_clf)
knn_accuracy = accuracy_score(y_test_clf, knn_predictions)
print("KNN Accuracy: {:.2f}%".format(knn_accuracy * 100))
print("KNN_predictions:")
print(knn_predictions)
print("-----knn_accuracy -----")
print(knn_accuracy ) #(-1-1)

#print("\nClassification  for KNN:")
#print(classification_report(y_test_clf, knn_predictions))
```

```
KNN Accuracy: 100.00%
KNN_predictions:
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
-----knn_accuracy -----
1.0
```

- **Decition Tree:**

In [65]:
```
print("Decision_tree :")
dt = DecisionTreeClassifier(random_state=0)
dt.fit(X_train_clf, y_train_clf)
dt_predictions = dt.predict(X_test_clf)
dt_accuracy = accuracy_score(y_test_clf, dt_predictions)
print("Decision Tree Accuracy: {:.2f}%".format(dt_accuracy * 100))
print("----dt_predictions-----")
print(dt_predictions)
print("-------dt_accuracy------")
print(dt_accuracy)#(-1-1)
```

```
Decision_tree :
Decision Tree Accuracy: 100.00%
----dt_predictions-----
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
-------dt_accuracy------
1.0
```

- **Naive Bayes :**

```
print("Naive Bayes :")
nb = GaussianNB()
nb.fit(X_train_clf, y_train_clf)
nb_predictions = nb.predict(X_test_clf)
nb_accuracy = accuracy_score(y_test_clf, nb_predictions)
print("Naive Bayes Accuracy: {:.2f}%".format(nb_accuracy * 100))
print('--------nb_predictions-------')
print(nb_predictions)
print("-----nb_accuracy-----")
print(nb_accuracy)
```

```
Naive Bayes :
Naive Bayes Accuracy: 50.00%
--------nb_predictions-------
[0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1]
-----nb_accuracy-----
0.5
```

- **Logistic regression**

```
print("\nLogistic Regression:")
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train_clf, y_train_clf)
log_reg_predictions = log_reg.predict(X_test_clf)
log_reg_accuracy = accuracy_score(y_test_clf, log_reg_predictions)
print("Logistic Regression Accuracy: {:.2f}%".format(log_reg_accuracy * 100))
```

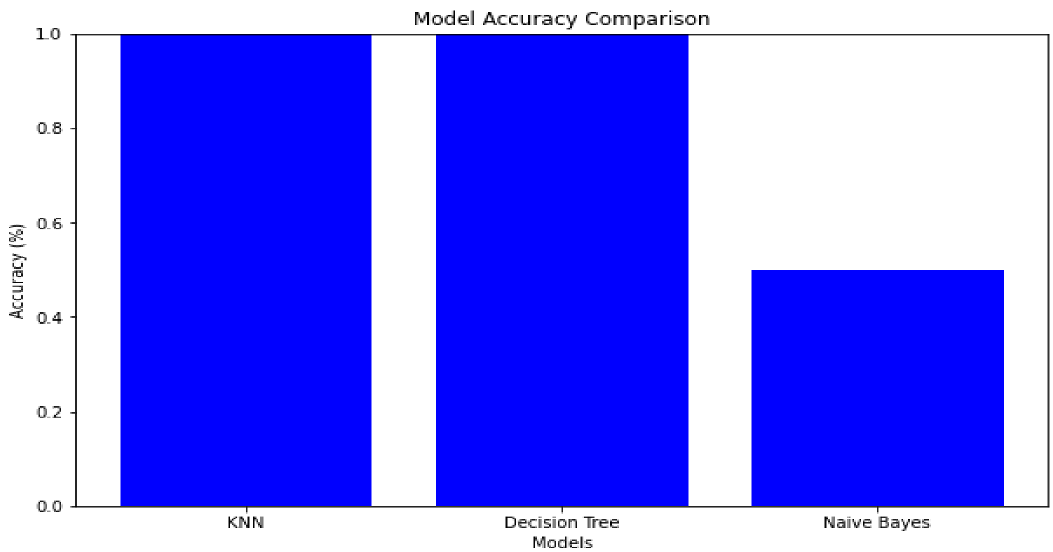
```
Logistic Regression:
Logistic Regression Accuracy: 100.00%
```

Model Selection/Comparism Analysis:

- **Barchart**

In [37]:
```python
models = ['KNN', 'Decision Tree', 'Naive Bayes']
accuracies = [knn_accuracy, dt_accuracy, nb_accuracy]
```

In [38]:
```python
#Bar-chart
plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color='blue')
plt.title('Model Accuracy Comparison')
plt.ylabel('Accuracy (%)')
plt.xlabel('Models')
plt.ylim(0, 1)
plt.show()
```



- **Precision Recall**

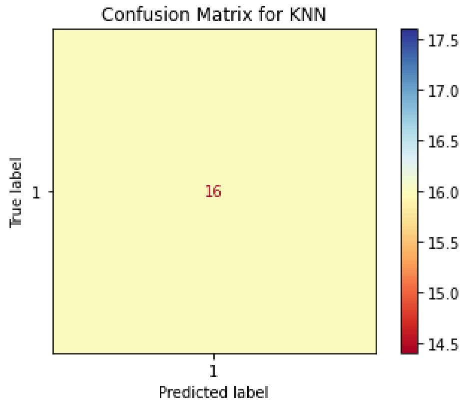
KNN Decision Tree Naive Bayes
Models

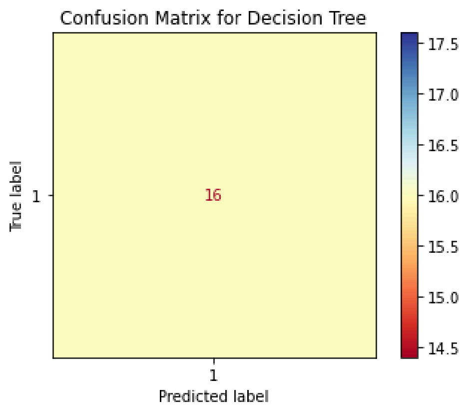
In [30]:
```python
#precition recall
from sklearn.metrics import precision_recall_fscore_support

for model_name, predictions in zip(models[:-1], [knn_predictions, dt_predictions, nb_predictions, log
    precision, recall, f1, _ = precision_recall_fscore_support(y_test_clf, predictions, average='binary')
    print(f"{model_name} -> Precision: {precision:.2f}, Recall: {recall:.2f}, F1-Score: {f1:.2f}")
    print(len(y_test_clf), len(predictions))
```

```
KNN -> Precision: 1.00, Recall: 1.00, F1-Score: 1.00
16 16
Decision Tree -> Precision: 1.00, Recall: 1.00, F1-Score: 1.00
16 16
Naive Bayes -> Precision: 1.00, Recall: 0.50, F1-Score: 0.67
16 16
```

- **Confusion matrix:**

In [91]:
```
# Confusion Matrix for each model
for model_name, predictions in zip(models[:-1], [knn_predictions, dt_predictions, nb_predictions ]):
    cm = confusion_matrix(y_test_clf, predictions, labels=np.unique(y_test_clf))
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.unique(y_test_clf))
    disp.plot(cmap='RdYlBu')  #rgb
    plt.title(f"Confusion Matrix for {model_name}")
    plt.show()
```





Conclusion:

- **Best performing model:** The best result-giving modes are KNN and decision tree (KNN -> Precision: 1.00, Recall: 1.00, F1-Score: 1.00)
  16 16
  Decision Tree: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
  16 16)
  This research shows the value of pre-processing and model selection for obtaining close to accurate predictions for both classification and regression tasks.