```
#Import some libraries to perform some calculations, visualization, plotting, remove warnings and other usage of functions

import pandas as pd
import matplotlib.pyplot as plt
import seaborn  as sns
import sklearn
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
#Load the dataset of Student and stored in variable called stu:

stu = pd.read_csv("/content/dataset.csv")
stu
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification	Mother's occupation	...	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	Curricular units 2nd sem (evaluations)	Curr uni (app
0	1	8	5	2	1	1	1	13	10	6	...	0	0	0	
1	1	6	1	11	1	1	1	1	3	4	...	0	6	6	
2	1	1	5	5	1	1	1	22	27	10	...	0	6	0	
3	1	8	2	15	1	1	1	23	27	6	...	0	6	10	
4	2	12	1	3	0	1	1	22	28	10	...	0	6	6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4419	1	1	6	15	1	1	1	1	1	6	...	0	6	8	
4420	1	1	2	15	1	1	19	1	1	10	...	0	6	6	
4421	1	1	1	12	1	1	1	22	27	10	...	0	8	9	
4422	1	1	1	9	1	1	1	22	27	8	...	0	5	6	
4423	1	5	1	15	1	1	9	23	27	6	...	0	6	6	

4424 rows × 35 columns

Basic Pandas

```
#This command gives the information of given dataset:

stu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 35 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Marital status                                4424 non-null   int64
 1   Application mode                              4424 non-null   int64
 2   Application order                             4424 non-null   int64
 3   Course                                        4424 non-null   int64
 4   Daytime/evening attendance                    4424 non-null   int64
 5   Previous qualification                        4424 non-null   int64
 6   Nacionality                                   4424 non-null   int64
 7   Mother's qualification                        4424 non-null   int64
 8   Father's qualification                        4424 non-null   int64
 9   Mother's occupation                           4424 non-null   int64
 10  Father's occupation                           4424 non-null   int64
 11  Displaced                                     4424 non-null   int64
 12  Educational special needs                     4424 non-null   int64
 13  Debtor                                        4424 non-null   int64
 14  Tuition fees up to date                       4424 non-null   int64
 15  Gender                                        4424 non-null   int64
 16  Scholarship holder                            4424 non-null   int64
 17  Age at enrollment                             4424 non-null   int64
 18  International                                 4424 non-null   int64
 19  Curricular units 1st sem (credited)           4424 non-null   int64
 20  Curricular units 1st sem (enrolled)           4424 non-null   int64
 21  Curricular units 1st sem (evaluations)        4424 non-null   int64
 22  Curricular units 1st sem (approved)           4424 non-null   int64
 23  Curricular units 1st sem (grade)              4424 non-null   float64
 24  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 25  Curricular units 2nd sem (credited)           4424 non-null   int64
 26  Curricular units 2nd sem (enrolled)           4424 non-null   int64
 27  Curricular units 2nd sem (evaluations)        4424 non-null   int64
 28  Curricular units 2nd sem (approved)           4424 non-null   int64
 29  Curricular units 2nd sem (grade)              4424 non-null   float64
 30  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 31  Unemployment rate                             4424 non-null   float64
 32  Inflation rate                                4424 non-null   float64
 33  GDP                                           4424 non-null   float64
 34  Target                                        4424 non-null   object
dtypes: float64(5), int64(29), object(1)
memory usage: 1.2+ MB
```

```
#This command gives the static information of given dataset:

stu.describe()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	q
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	
mean	1.178571	6.886980	1.727848	9.899186	0.890823	2.531420	1.254521	12.322107	
std	0.605747	5.298964	1.313793	4.331792	0.311897	3.963707	1.748447	9.026251	
min	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000	
25%	1.000000	1.000000	1.000000	6.000000	1.000000	1.000000	1.000000	2.000000	
50%	1.000000	8.000000	1.000000	10.000000	1.000000	1.000000	1.000000	13.000000	
75%	1.000000	12.000000	2.000000	13.000000	1.000000	1.000000	1.000000	22.000000	
max	6.000000	18.000000	9.000000	17.000000	1.000000	17.000000	21.000000	29.000000	

8 rows × 34 columns

```
#This command shows the first 5 rows of given dataset:

stu.head()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Father's qualification
0	1	8	5	2	1	1	1	13	10
1	1	6	1	11	1	1	1	1	3
2	1	1	5	5	1	1	1	22	27
3	1	8	2	15	1	1	1	23	27
4	2	12	1	3	0	1	1	22	28

5 rows × 35 columns

```
#This command shows the last 5 rows of given dataset:

stu.tail()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Fathe qualificat
4419	1	1	6	15	1	1	1	1	
4420	1	1	2	15	1	1	19	1	
4421	1	1	1	12	1	1	1	22	
4422	1	1	1	9	1	1	1	22	
4423	1	5	1	15	1	1	9	23	

5 rows × 35 columns

```
#This command shows the columns of given dataset:

stu.columns
```

```
Index(['Marital status', 'Application mode', 'Application order', 'Course',
       'Daytime/evening attendance', 'Previous qualification', 'Nacionality',
       'Mother's qualification', 'Father's qualification',
       'Mother's occupation', 'Father's occupation', 'Displaced',
       'Educational special needs', 'Debtor', 'Tuition fees up to date',
       'Gender', 'Scholarship holder', 'Age at enrollment', 'International',
       'Curricular units 1st sem (credited)',
       'Curricular units 1st sem (enrolled)',
       'Curricular units 1st sem (evaluations)',
       'Curricular units 1st sem (approved)',
       'Curricular units 1st sem (grade)',
       'Curricular units 1st sem (without evaluations)',
       'Curricular units 2nd sem (credited)',
       'Curricular units 2nd sem (enrolled)',
       'Curricular units 2nd sem (evaluations)',
       'Curricular units 2nd sem (approved)',
       'Curricular units 2nd sem (grade)',
       'Curricular units 2nd sem (without evaluations)', 'Unemployment rate',
       'Inflation rate', 'GDP', 'Target'],
      dtype='object')
```

```
#This command shows the order pair of given dataset:

stu.shape
```

```
(4424, 35)
```

```
#This command gives the duplicated values  of given dataset:

stu.duplicated().sum()
```

```
0
```

```
#This command gives the missing values of given dataset:

stu.isnull().sum()
```

```
Marital status                                  0
Application mode                                0
Application order                               0
Course                                          0
Daytime/evening attendance                      0
Previous qualification                          0
Nacionality                                     0
Mother's qualification                          0
Father's qualification                          0
Mother's occupation                             0
Father's occupation                             0
Displaced                                       0
Educational special needs                       0
Debtor                                          0
Tuition fees up to date                         0
Gender                                          0
Scholarship holder                              0
Age at enrollment                               0
International                                   0
Curricular units 1st sem (credited)             0
Curricular units 1st sem (enrolled)             0
Curricular units 1st sem (evaluations)          0
Curricular units 1st sem (approved)             0
Curricular units 1st sem (grade)                0
Curricular units 1st sem (without evaluations)  0
Curricular units 2nd sem (credited)             0
Curricular units 2nd sem (enrolled)             0
Curricular units 2nd sem (evaluations)          0
Curricular units 2nd sem (approved)             0
Curricular units 2nd sem (grade)                0
Curricular units 2nd sem (without evaluations)  0
Unemployment rate                               0
Inflation rate                                  0
GDP                                             0
Target                                          0
dtype: int64
```

```
#This command counts the values of target column:

stu.Target.value_counts()
```

```
Graduate    2209
Dropout     1421
Enrolled     794
Name: Target, dtype: int64
```

```
#This command counts the values of GDP column:

stu.GDP.value_counts()
```

```
 0.32    571
-3.12    533
 1.74    525
 1.79    445
-1.70    419
 2.02    414
-4.06    397
 0.79    390
 3.51    368
-0.92    362
Name: GDP, dtype: int64
```

```
#This command counts the values of Course column:

stu.Course.value_counts()
```

```
12    766
9     380
10    355
6     337
15    331
14    268
17    268
11    252
5     226
2     215
3     215
4     210
16    192
7     170
8     141
13     86
1      12
Name: Course, dtype: int64
```

```
#This command gives the data type of all columns:
data_types = stu.dtypes
print("Data Types:\n", data_types)
```

```
Data Types:
 Marital status                                  int64
Application mode                                 int64
Application order                                int64
Course                                           int64
Daytime/evening attendance                       int64
Previous qualification                           int64
Nacionality                                      int64
Mother's qualification                           int64
Father's qualification                           int64
Mother's occupation                              int64
Father's occupation                              int64
Displaced                                        int64
Educational special needs                        int64
Debtor                                           int64
Tuition fees up to date                          int64
Gender                                           int64
Scholarship holder                               int64
Age at enrollment                                int64
International                                    int64
Curricular units 1st sem (credited)              int64
Curricular units 1st sem (enrolled)              int64
Curricular units 1st sem (evaluations)           int64
Curricular units 1st sem (approved)              int64
Curricular units 1st sem (grade)               float64
Curricular units 1st sem (without evaluations)   int64
Curricular units 2nd sem (credited)              int64
Curricular units 2nd sem (enrolled)              int64
Curricular units 2nd sem (evaluations)           int64
Curricular units 2nd sem (approved)              int64
Curricular units 2nd sem (grade)               float64
Curricular units 2nd sem (without evaluations)   int64
Unemployment rate                              float64
Inflation rate                                 float64
GDP                                            float64
Target                                          object
dtype: object
```

```
#This command groups the target column ("is_claim") into all other columns.
#Following cammand shows the relationship between target and other column:

gp = stu.groupby('Target').count()
gp
```

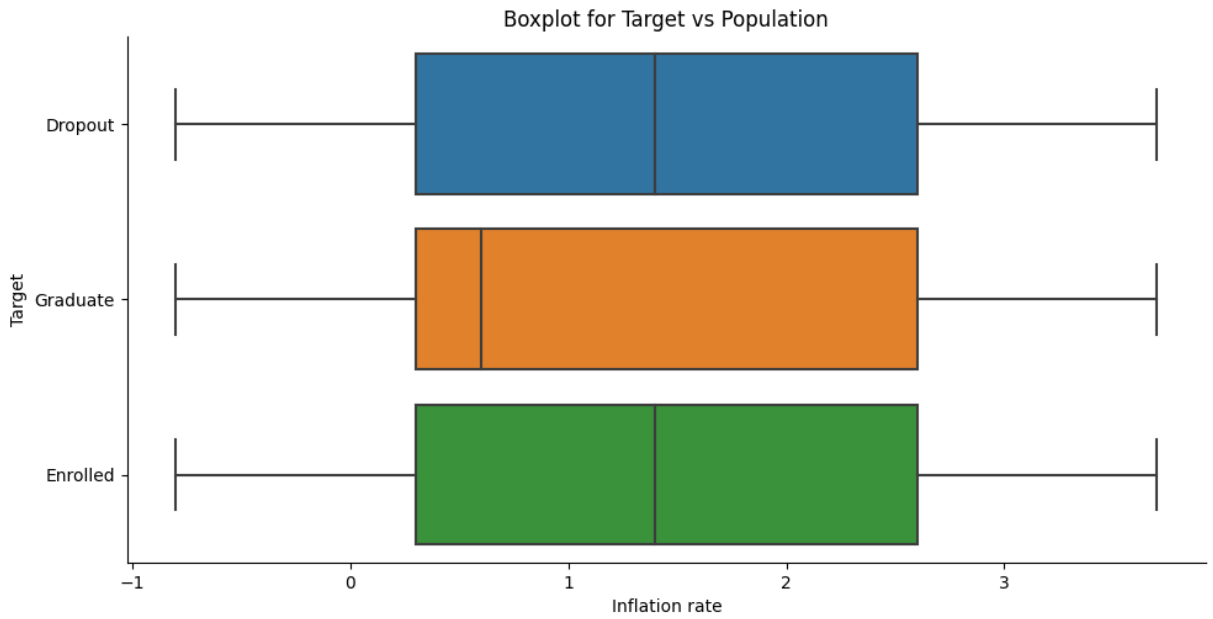
Target	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother's qualification	Fa qualifi
Dropout	1421	1421	1421	1421	1421	1421	1421	1421	
Enrolled	794	794	794	794	794	794	794	794	
Graduate	2209	2209	2209	2209	2209	2209	2209	2209	

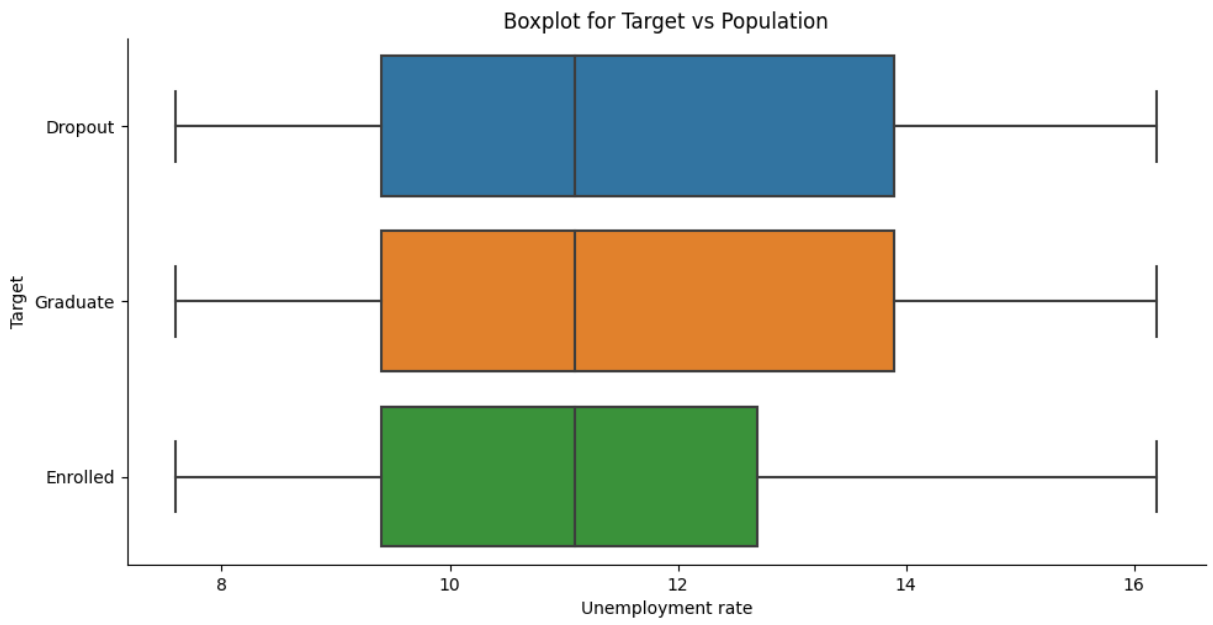
3 rows × 34 columns

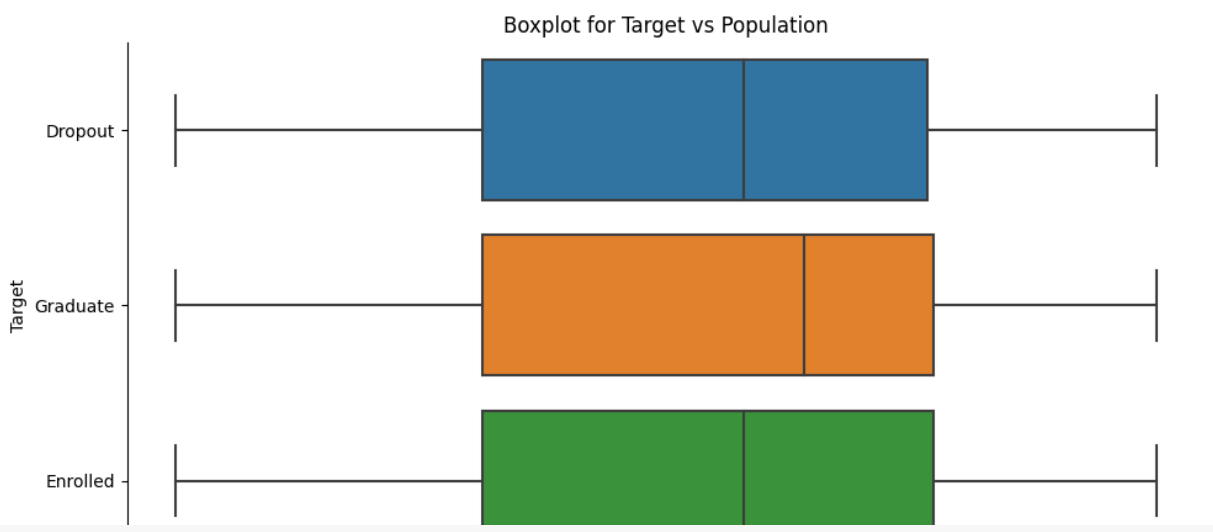
Visualization

```
#Tis command is used to draw a catplot between target column and other columns:

num_col = ['Inflation rate','Unemployment rate','GDP']
for i in num_col:
  sns.catplot(x= i, y ="Target", data =stu, kind = "box", aspect = 2)
  plt.title("Boxplot for Target vs Population")
  plt.show()
```

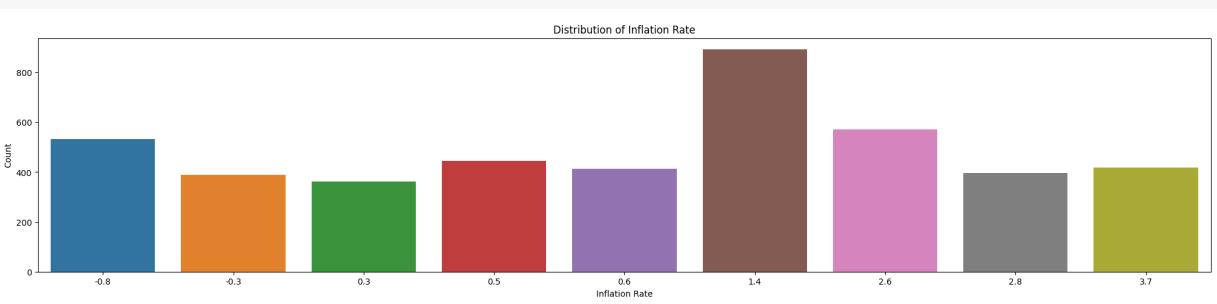






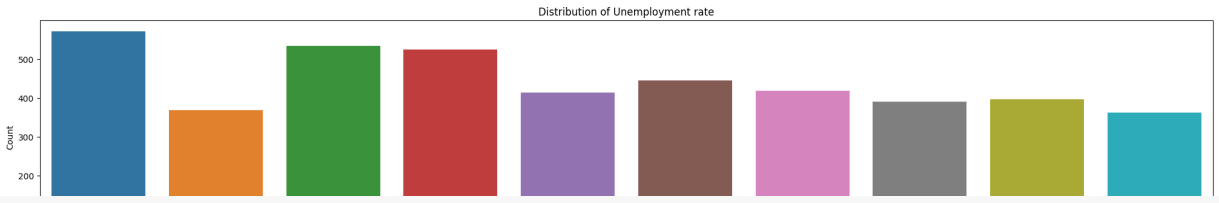

```
#This command is used to draw a count plot of Inflation rate column:

plt.figure(figsize = (25,5))
sns.countplot(x='Inflation rate', data=stu)
plt.xlabel('Inflation Rate')
plt.ylabel('Count')
plt.title('Distribution of Inflation Rate')
plt.show()
```



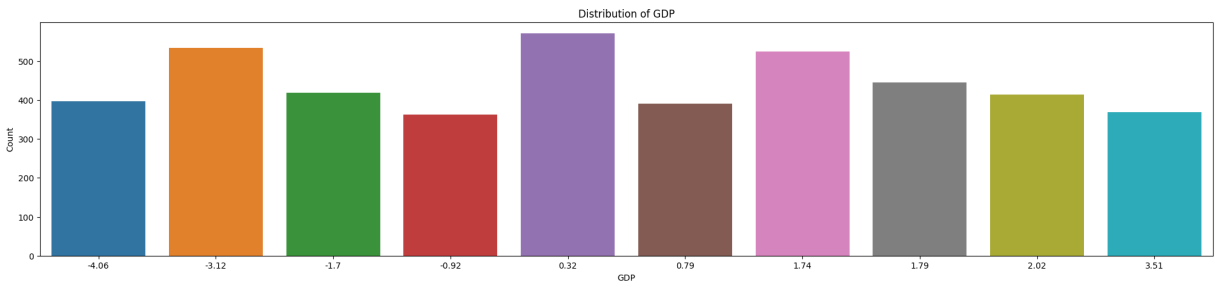

```
#This command is used to draw a count plot of Unemployment rate column:

plt.figure(figsize = (25,5))
sns.countplot(x='Unemployment rate', data=stu)
plt.xlabel('Unemployment rate')
plt.ylabel('Count')
plt.title('Distribution of Unemployment rate')
plt.show()
```



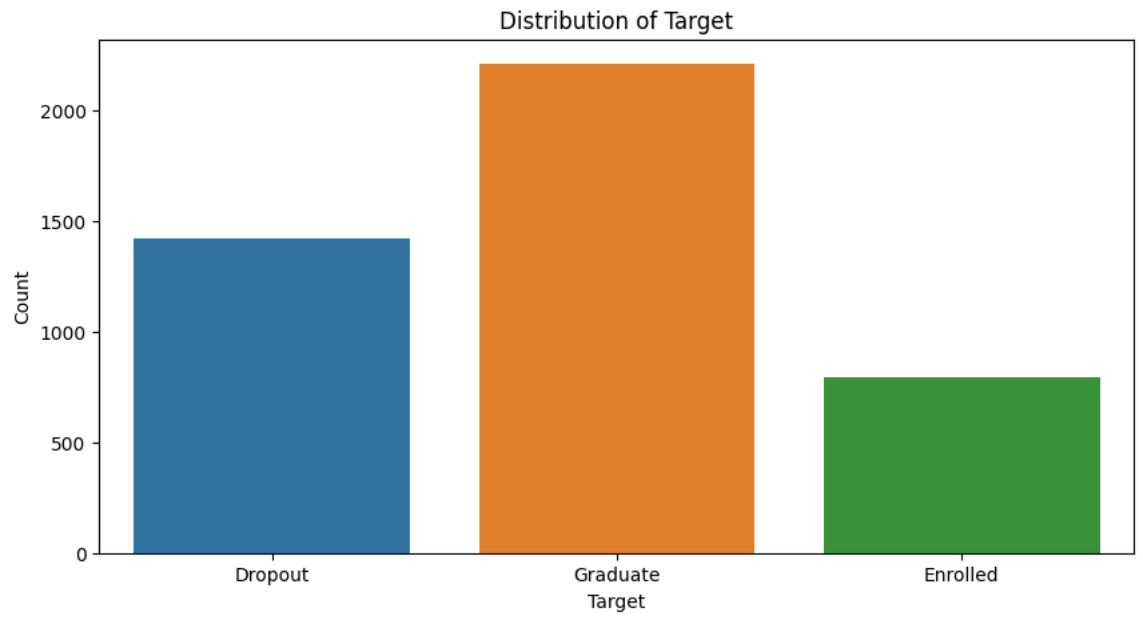

```
#This command is used to draw a count plot of GDP column:

plt.figure(figsize = (25,5))
sns.countplot(x='GDP', data=stu)
plt.xlabel('GDP')
plt.ylabel('Count')
plt.title('Distribution of GDP')
plt.show()
```




```
#This command is used to draw a count plot of Target column:

plt.figure(figsize = (10,5))
sns.countplot(x='Target', data=stu)
plt.xlabel('Target')
plt.ylabel('Count')
plt.title('Distribution of Target')
plt.show()
```



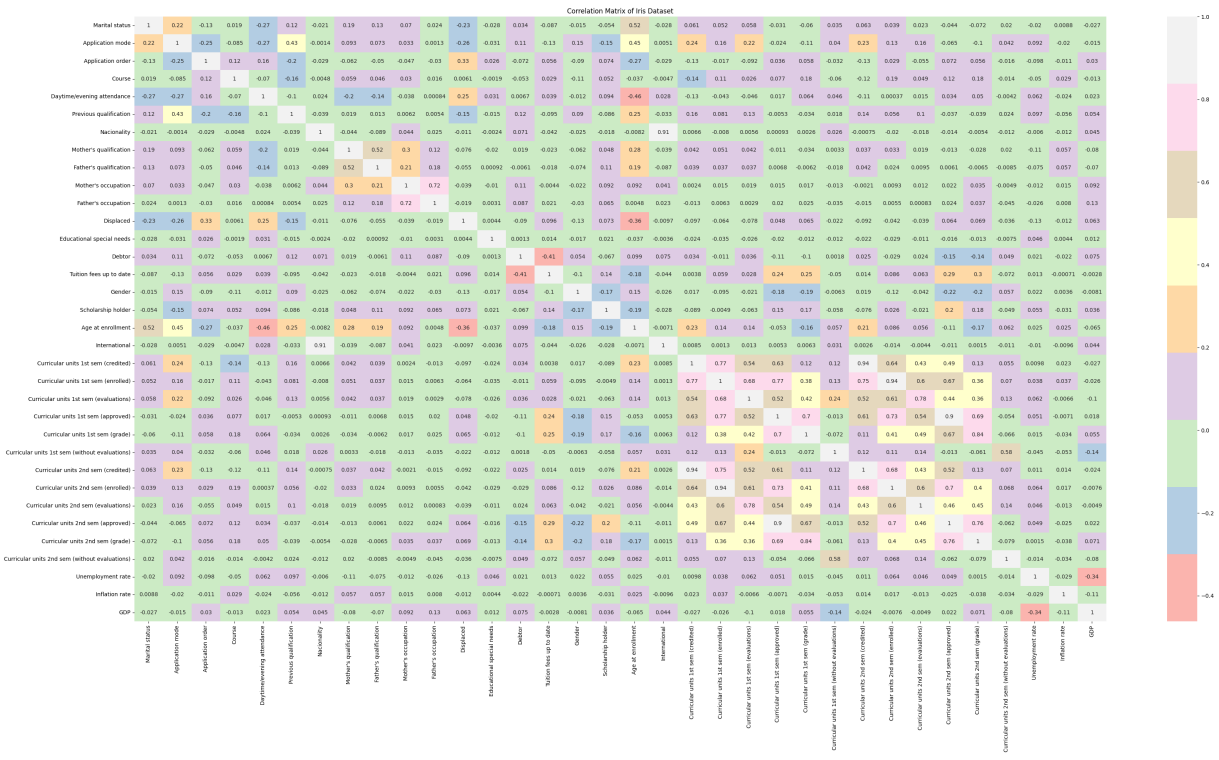

```
#This command is used to draw correlation matrix:

stu.corr()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Mother qualificatio
Marital status	1.000000	0.224855	-0.125854	0.018925	-0.274939	0.120925	-0.020722	0.1855
Application mode	0.224855	1.000000	-0.246497	-0.085116	-0.268616	0.433028	-0.001360	0.0928
Application order	-0.125854	-0.246497	1.000000	0.118928	0.158657	-0.199029	-0.029385	-0.0617
Course	0.018925	-0.085116	0.118928	1.000000	-0.070232	-0.158382	-0.004761	0.0589
Daytime/evening attendance	-0.274939	-0.268616	0.158657	-0.070232	1.000000	-0.103022	0.024433	-0.1953
Previous qualification	0.120925	0.433028	-0.199029	-0.158382	-0.103022	1.000000	-0.038997	0.0188
Nacionality	-0.020722	-0.001360	-0.029385	-0.004761	0.024433	-0.038997	1.000000	-0.0438
Mother's qualification	0.185522	0.092867	-0.061719	0.058909	-0.195346	0.018868	-0.043847	1.0000
Father's qualification	0.128326	0.072798	-0.049936	0.045659	-0.137769	0.013152	-0.088892	0.5245
Mother's occupation	0.069734	0.033489	-0.046591	0.029672	-0.037986	0.006190	0.044123	0.2951
Father's occupation	0.024351	0.001253	-0.029754	0.016489	0.000845	0.005381	0.024538	0.1159
Displaced	-0.234886	-0.263079	0.332362	0.006142	0.251767	-0.149356	-0.010774	-0.0758
Educational special needs	-0.028343	-0.030868	0.025597	-0.001886	0.031017	-0.015015	-0.002399	-0.0198
Debtor	0.034304	0.114348	-0.072151	-0.053149	0.006658	0.117447	0.070860	0.0187
Tuition fees up to date	-0.087158	-0.127339	0.055891	0.029099	0.038799	-0.095246	-0.041721	-0.0228
Gender	-0.014738	0.147226	-0.089559	-0.111383	-0.012326	0.089952	-0.025462	-0.0623
Scholarship holder	-0.053765	-0.152818	0.073709	0.051668	0.093912	-0.085668	-0.018468	0.0482
Age at enrollment	0.522717	0.450700	-0.271154	-0.036929	-0.462280	0.249821	-0.008241	0.2799
International	-0.027905	0.005050	-0.028801	-0.004662	0.027973	-0.033498	0.911724	-0.0386
Curricular units 1st sem (credited)	0.061209	0.238269	-0.133354	-0.140546	-0.127466	0.159940	0.006604	0.0416
Curricular units 1st sem	0.052107	0.150547	0.016808	0.112285	0.043056	0.080860	0.008011	0.0505

```
#This command is used to draw a heatmap of correlation matrix:

correlation_matrix = stu.corr()
plt.figure(figsize = (40,20))
sns.heatmap(correlation_matrix, annot=True, cmap= 'Pastel1')
plt.title('Correlation Matrix of Iris Dataset')
plt.show()
```



▾ *Label Encoding at Target column, Split data into train and test model*

```
#import libraries  accuracy and test-train slipt:

import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

```
# This command is used to Split dataset into X and y:

X = stu.drop(columns= ['Target'])
y = stu[['Target']]
```

```
#This command is used to show the order pair of X and y:

print("X shape is: ",X.shape)
print("y shape is: ",y.shape)
```

```
X shape is:  (4424, 34)
y shape is:  (4424, 1)
```

```
#This command split given dataset into test and train:

X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size = 0.20)
```

```
#This command shows the order pair of test and train

print("shape of X_train: ", X_train.shape)
print("shape of Y_train: ", Y_train.shape)
print("shape of X_test: ", X_test.shape)
print("shape of Y_test: ", Y_test.shape)
```

```
shape of X_train:  (3539, 34)
shape of Y_train:  (3539, 1)
shape of X_test:  (885, 34)
shape of Y_test:  (885, 1)
```

Classification Model --> "Logistic Regression"

```
#Import Logistic Regression libraries:

from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
# Fit the Model:

model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

```
#Calculate accuracy score:

y_pred = model.predict(X_test)
score = accuracy_score(Y_test, y_pred)
accuracy = score*100
print(accuracy)
```

```
78.19209039548024
```

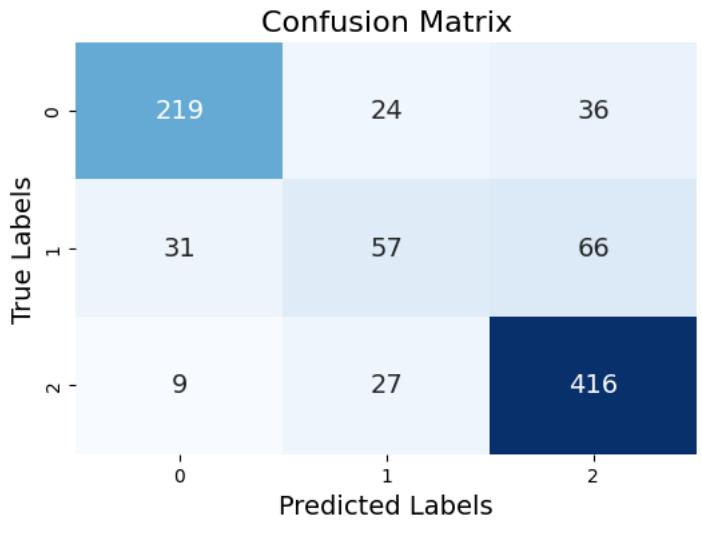
```
# This command is used to draw confussion matrix:

from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)
print("Confusion Matrix")
cm_lr = confusion_matrix(Y_test, y_pred)
print(cm_lr)
```

```
Confusion Matrix
[[219  24  36]
 [ 31  57  66]
 [  9  27 416]]
```

```
# This command is used to draw a heatmap of confussion matrix:

plt.figure(figsize = (6, 4))
sns.heatmap(cm_lr, annot = True, fmt = 'd', cmap = 'Blues', cbar = False, annot_kws = {'size' : 14})
plt.xlabel('Predicted Labels', fontsize = 14)
plt.ylabel('True Labels', fontsize = 14)
plt.title('Confusion Matrix', fontsize = 16)
plt.show()
```



```
#This command is used to verify the model based on given dataset:

X_test1 = X_test.values
y_test1 = Y_test.values
y_pred = str(model.predict([X_test1[16]]))
print(type(y_pred))

if y_pred == "[0]":
  print("Dropout")
elif y_pred == "[1]":
  print("Enrolled")
elif y_pred == "[2]":
  print("Graduate")
```

```
<class 'str'>
Graduate
```

Classification Model --> "Decision Tree"

```
#Import decesion tree classifier library and develop the model:

from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
```

```
# Fit model:

dtc.fit(X_train, Y_train)
```

```
DecisionTreeClassifier()
```

```
#This command is used to calculate accuracy of the model:

y_pred = dtc.predict(X_test)
score = accuracy_score(Y_test, y_pred)
accuracy = score*100
print(accuracy)
```

```
68.47457627118644
```

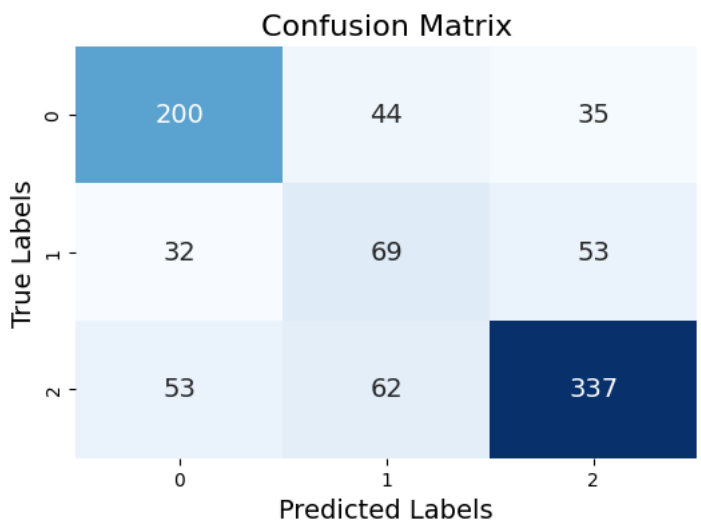
```
# This command is used to draw confussion matrix:

from sklearn.metrics import confusion_matrix
y_pred = dtc.predict(X_test)
print("Confusion Matrix")
cm_dtc = confusion_matrix(Y_test, y_pred)
print(cm_dtc)
```

```
Confusion Matrix
[[200  44  35]
 [ 32  69  53]
 [ 53  62 337]]
```

```
# This command is used to draw a heatmap of confussion matrix:

plt.figure(figsize = (6, 4))
sns.heatmap(cm_dtc, annot = True, fmt = 'd', cmap = 'Blues', cbar = False, annot_kws = {'size' : 14})
plt.xlabel('Predicted Labels', fontsize = 14)
plt.ylabel('True Labels', fontsize = 14)
plt.title('Confusion Matrix', fontsize = 16)
plt.show()
```



Classification Model --> "Support Vector Machine (SVM)"

```
#Import support vector machine library and develop the model:

from sklearn.svm import SVC
sss = SVC()
```

```
#Fit model:

sss.fit(X_train, Y_train)
```

```
SVC()
```

```
#This command is used to calculate accuracy of the model:

y_pred = sss.predict(X_test)
score = accuracy_score(Y_test, y_pred)
accuracy = score*100
print(accuracy)
```

```
74.23728813559322
```

```
# This command is used to draw confussion matrix:

from sklearn.metrics import confusion_matrix
y_pred = sss.predict(X_test)
print("Confusion Matrix")
cm_sss = confusion_matrix(Y_test, y_pred)
print(cm_sss)
```

```
Confusion Matrix
[[195  34  50]
 [ 32  42  80]
 [ 15  17 420]]
```

```
# This command is used to draw a heatmap of confussion matrix:

plt.figure(figsize = (6, 4))
sns.heatmap(cm_sss, annot = True, fmt = 'd', cmap = 'Blues', cbar = False, annot_kws = {'size' : 14})
plt.xlabel('Predicted Labels', fontsize = 14)
plt.ylabel('True Labels', fontsize = 14)
plt.title('Confusion Matrix', fontsize = 16)
plt.show()
```