FINAL v2

October 7, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv(
    "/Users/gabrielmancillas/Documents/GitHub/StudentPerformancePrediction/
 ↪dataset.csv"
)
```

[2]:
```
df.head()
```

[2]:

	Marital status	Application mode	Application order	Course	\
0	1	8	5	2	
1	1	6	1	11	
2	1	1	5	5	
3	1	8	2	15	
4	2	12	1	3	

	Daytime/evening attendance	Previous qualification	Nacionality	\
0	1	1	1	
1	1	1	1	
2	1	1	1	
3	1	1	1	
4	0	1	1	

	Mother's qualification	Father's qualification	Mother's occupation	…	\
0	13	10	6	…	
1	1	3	4	…	
2	22	27	10	…	
3	23	27	6	…	
4	22	28	10	…	

	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	\
0	0	0	
1	0	6	
2	0	6	
3	0	6	

```
4                                     0                                     6

   Curricular units 2nd sem (evaluations)  \
0                                       0
1                                       6
2                                       0
3                                      10
4                                       6

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)  \
0                                    0                          0.000000
1                                    6                         13.666667
2                                    0                          0.000000
3                                    5                         12.400000
4                                    6                         13.000000

   Curricular units 2nd sem (without evaluations)  Unemployment rate  \
0                                               0               10.8
1                                               0               13.9
2                                               0               10.8
3                                               0                9.4
4                                               0               13.9

   Inflation rate   GDP    Target
0             1.4  1.74   Dropout
1            -0.3  0.79  Graduate
2             1.4  1.74   Dropout
3            -0.8 -3.12  Graduate
4            -0.3  0.79  Graduate

[5 rows x 35 columns]
```

[3]:
```
# numbers of students
df.shape
```

[3]:
```
(4424, 35)
```

[4]:
```
df.rename(columns={"Nacionality": "Nationality"}, inplace=True)
```

[5]:
```
df.describe().round(3)
```

[5]:
```
       Marital status  Application mode  Application order    Course  \
count        4424.000          4424.000           4424.000  4424.000
mean            1.179             6.887              1.728     9.899
std             0.606             5.299              1.314     4.332
min             1.000             1.000              0.000     1.000
25%             1.000             1.000              1.000     6.000
```

```
50%             1.000             8.000             1.000    10.000
75%             1.000            12.000             2.000    13.000
max             6.000            18.000             9.000    17.000

       Daytime/evening attendance  Previous qualification  Nationality  \
count                    4424.000                4424.000     4424.000
mean                        0.891                   2.531        1.255
std                         0.312                   3.964        1.748
min                         0.000                   1.000        1.000
25%                         1.000                   1.000        1.000
50%                         1.000                   1.000        1.000
75%                         1.000                   1.000        1.000
max                         1.000                  17.000       21.000

       Mother's qualification  Father's qualification  Mother's occupation  \
count                4424.000                4424.000             4424.000
mean                   12.322                  16.455                7.318
std                     9.026                  11.045                3.998
min                     1.000                   1.000                1.000
25%                     2.000                   3.000                5.000
50%                    13.000                  14.000                6.000
75%                    22.000                  27.000               10.000
max                    29.000                  34.000               32.000

       …  Curricular units 1st sem (without evaluations)  \
count  …                                        4424.000
mean   …                                           0.138
std    …                                           0.691
min    …                                           0.000
25%    …                                           0.000
50%    …                                           0.000
75%    …                                           0.000
max    …                                          12.000

       Curricular units 2nd sem (credited)  \
count                              4424.000
mean                                  0.542
std                                   1.919
min                                   0.000
25%                                   0.000
50%                                   0.000
75%                                   0.000
max                                  19.000

       Curricular units 2nd sem (enrolled)  \
count                              4424.000
mean                                  6.232
```

std	2.196
min	0.000
25%	5.000
50%	6.000
75%	7.000
max	23.000

	Curricular units 2nd sem (evaluations)
count	4424.000
mean	8.063
std	3.948
min	0.000
25%	6.000
50%	8.000
75%	10.000
max	33.000

	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)
count	4424.000	4424.000
mean	4.436	10.230
std	3.015	5.211
min	0.000	0.000
25%	2.000	10.750
50%	5.000	12.200
75%	6.000	13.333
max	20.000	18.571

	Curricular units 2nd sem (without evaluations)	Unemployment rate
count	4424.000	4424.000
mean	0.150	11.566
std	0.754	2.664
min	0.000	7.600
25%	0.000	9.400
50%	0.000	11.100
75%	0.000	13.900
max	12.000	16.200

	Inflation rate	GDP
count	4424.000	4424.000
mean	1.228	0.002
std	1.383	2.270
min	-0.800	-4.060
25%	0.300	-1.700
50%	1.400	0.320
75%	2.600	1.790
max	3.700	3.510

```
[8 rows x 34 columns]
```

[6]:
```
df.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   Nationality                                     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
```

[7]:
```
print(df.isna().sum())
print("Total Missing: ", df.isna().sum().sum())
```

```
Marital status                                  0
Application mode                                0
Application order                               0
Course                                          0
Daytime/evening attendance                      0
Previous qualification                          0
Nationality                                     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
Total Missing:  0
```

we are working zero missing values

[8]:
```
print("Total Duplicates: ", df.duplicated().sum())
```

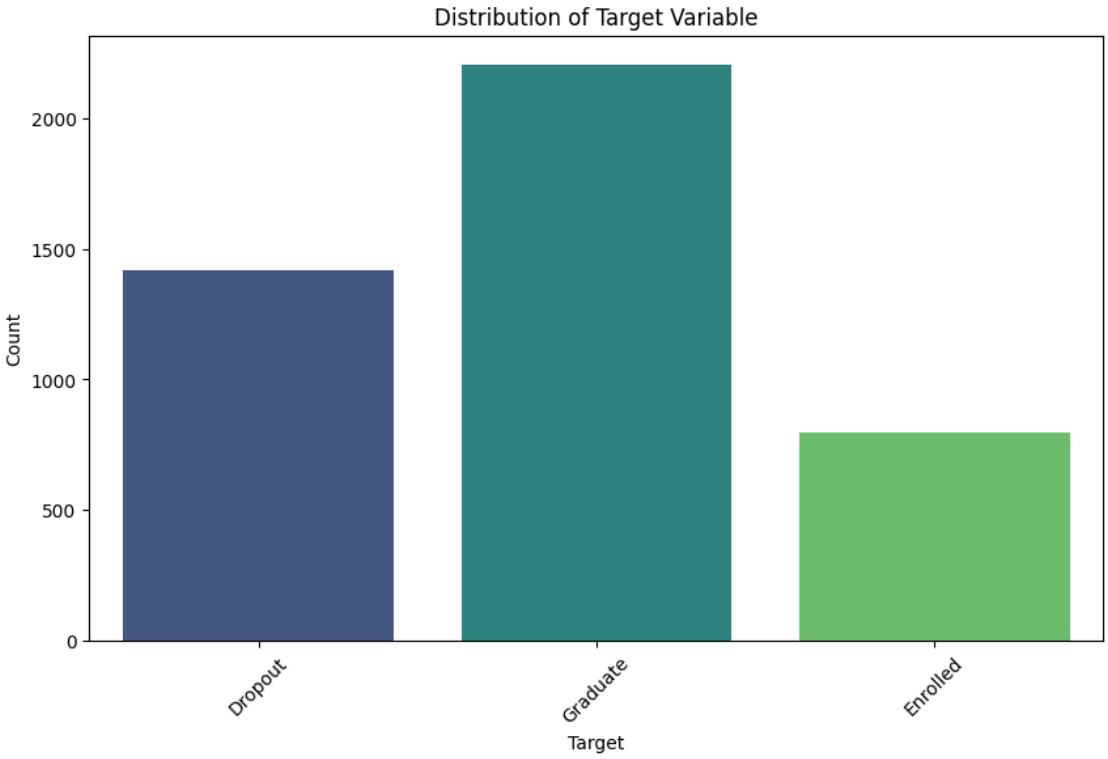
```
Total Duplicates:  0
```

[9]:
```
df["Target"].value_counts()
```

[9]:
```
Target
Graduate    2209
Dropout     1421
Enrolled     794
Name: count, dtype: int64
```

[10]:
```
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x="Target", palette="viridis")
plt.title("Distribution of Target Variable")
plt.xlabel("Target")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.show()
```

```
/var/folders/jw/4t4swxld5c5f_5xhv0_bzbr00000gn/T/ipykernel_21503/3590150.py:2:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df, x='Target', palette='viridis')
```



[11]:
```
df = df[df.Target != "Enrolled"]
```

[12]:
```
df.shape
```

[12]:
```
(3630, 35)
```

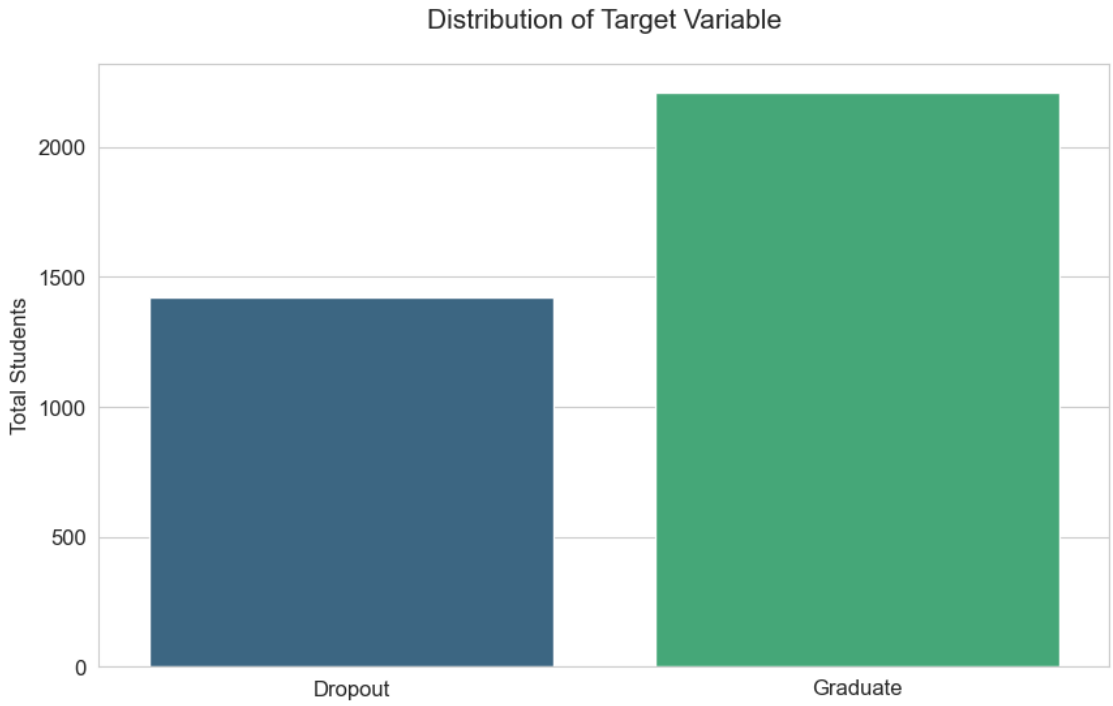
[13]:
```
freq_distribution = df["Target"].value_counts().to_frame(name="Count")
freq_distribution["% of Total"] = (
    df["Target"].value_counts(normalize=True) * 100
).round(2)
freq_distribution
```

[13]:

Target	Count	% of Total
Graduate	2209	60.85
Dropout	1421	39.15

[14]:
```
sns.set_style("whitegrid")
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x="Target", palette="viridis")  # Changed palette to␣
↪'viridis'

plt.ylabel("Total Students", fontsize=12)
plt.xlabel(None)
plt.title("Distribution of Target Variable", pad=20, fontsize=15)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

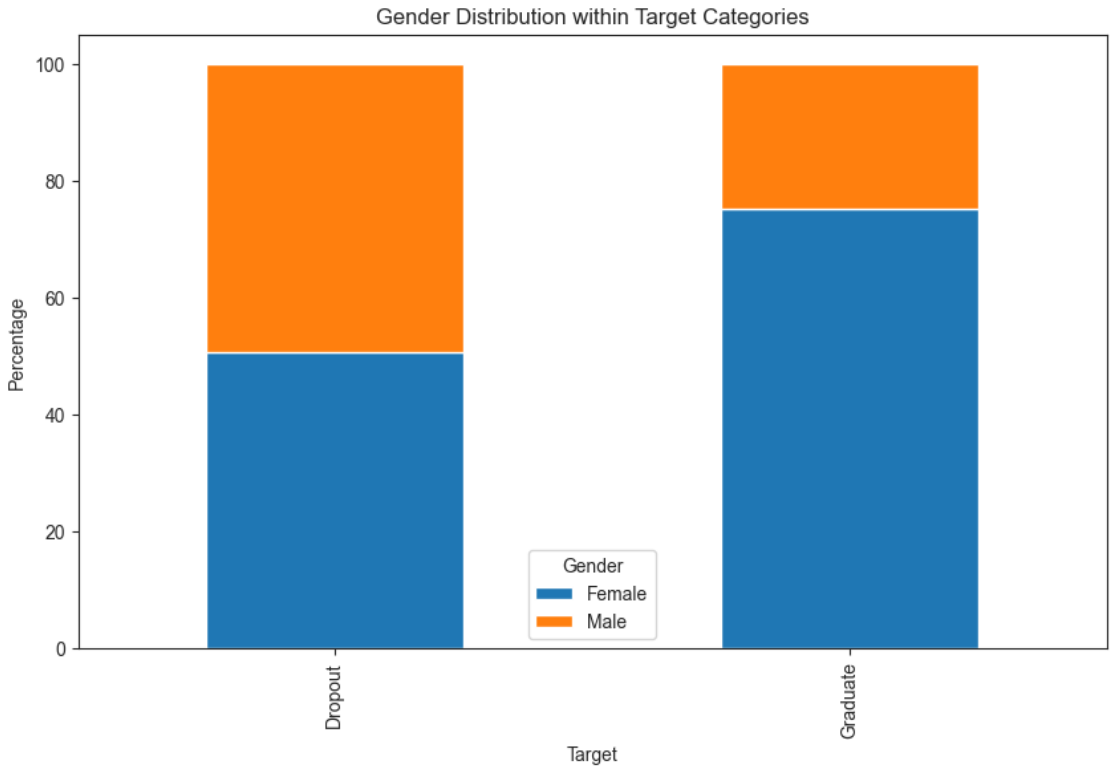
```
/var/folders/jw/4t4swxld5c5f_5xhv0_bzbr00000gn/T/ipykernel_21503/1649841069.py:3
: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(data=df, x='Target', palette='viridis')  # Changed palette to
'viridis'
```



[15]:
```
sns.set_style("ticks")
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x="Gender", hue="Target", palette="viridis")

plt.xticks(ticks=[0, 1], labels=["Female", "Male"])
plt.ylabel("Total Students", fontsize=12)
plt.xlabel(None)
plt.title("Distribution of Target by Gender", pad=20, fontsize=15)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

plt.show()
```

[16]:
```
# Calculate the crosstab of Target and Gender
ct_gender = pd.crosstab(df["Target"], df["Gender"])

# Rename columns for better readability
ct_gender.columns = ["Female", "Male"]

# Calculate the percentage distribution within each Target category
ct_gender_percentage = ct_gender.div(ct_gender.sum(axis=1), axis=0) * 100

# Display the crosstab with counts and percentages
ct_gender_combined = ct_gender.copy()
ct_gender_combined["Female (%)"] = ct_gender_percentage["Female"]
ct_gender_combined["Male (%)"] = ct_gender_percentage["Male"]

# Add a column for the total percentage
ct_gender_combined["Total (%)"] = (
    ct_gender_combined["Female (%)"] + ct_gender_combined["Male (%)"]
)
ct_gender_combined

# Plot the percentage distribution using a stacked bar plot
ct_gender_percentage.plot(
    kind="bar", stacked=True, figsize=(10, 6), color=["#1f77b4", "#ff7f0e"]
)
```

```
plt.title("Gender Distribution within Target Categories")
plt.xlabel("Target")
plt.ylabel("Percentage")
plt.legend(title="Gender")
plt.show()
```



[17]:
```
import plotly.express as px

# Create an interactive histogram with more bins
fig = px.histogram(
    df,
    x="Age at enrollment",
    nbins=30,
    title="Distribution by Age",
    labels={"Age at enrollment": "Age at Enrollment", "count": "Total␣
 ↪Students"},
    color_discrete_sequence=["dodgerblue"],
)

# Customize the layout
fig.update_layout(
    title={"text": "Distribution by Age", "x": 0.5},
```

```
    xaxis_title="Age at Enrollment",
    yaxis_title="Total Students",
    bargap=0.1,
)

# Show the plot
fig.show()
```

[24]:
```
# Create an interactive count plot
fig = px.histogram(
    df,
    x="Marital status",
    color="Target",
    barmode="group",
    title="Distribution of Target by Marital Status",
    labels={"Marital status": "Marital Status", "count": "Total Students"},
    color_discrete_sequence=["dodgerblue", "orange"],
)

# Customize the layout
fig.update_layout(
    title={"text": "Distribution of Target by Marital Status", "x": 0.5},
    xaxis_title="Marital Status",
    yaxis_title="Total Students",
    bargap=0.1,
)

# Change the x tick labels to the corresponding status
fig.update_xaxes(
    tickvals=[1, 2, 3, 4, 5, 6],
    ticktext=[
        "Single",
        "Married",
        "Widower",
        "Divorced",
        "Defacto union",
        "Legally separated",
    ],
)

# Show the plot
fig.show()
```

[47]:
```
import plotly.express as px

# Group by Course and Target
student_courses = (
```

```
    df.groupby(["Course", "Target"])
    .size()
    .reset_index()
    .pivot(columns="Target", index="Course", values=0)
)
student_courses = student_courses.rename(
    index={
        1: "Biofuel Production Technologies",
        2: "Animation and Multimedia Design",
        3: "Social Service (evening attendance)",
        4: "Agronomy",
        5: "Communication Design",
        6: "Veterinary Nursing",
        7: "Informatics Engineering",
        8: "Equinculture",
        9: "Management",
        10: "Social Service",
        11: "Tourism",
        12: "Nursing",
        13: "Oral Hygiene",
        14: "Advertising and Marketing Management",
        15: "Journalism and Communication",
        16: "Basic Education",
        17: "Management (evening attendance)",
    }
)

# Ensure the 'Dropout' column exists
if "Dropout" not in student_courses.columns:
    student_courses["Dropout"] = student_courses[0]  # Assuming '0' represents␣
 ↪dropouts

# Sum the total number of students for each course and sort for the plot
student_courses["Total"] = student_courses.sum(axis=1)
student_courses_sorted = student_courses.sort_values(by="Total", ascending=True)

# Remove the 'Total' column
student_courses_sorted.drop(columns="Total", inplace=True)

# Generate the interactive plot
fig = px.bar(
    student_courses_sorted,
    orientation="h",
    title="Distribution of Target by Course",
    labels={"value": "Total Students", "Course": "Course"},
    color_discrete_sequence=px.colors.qualitative.Pastel,
)
```

```
# Customize the layout
fig.update_layout(
    title={"text": "<b>Distribution of Target by Course</b>", "x": 0.5},
    xaxis_title="<b>Total Students</b>",
    yaxis_title=None,
    barmode="stack",
    width=1200,  # Increase the width
    height=800,  # Increase the height
)

# Show the plot
fig.show()
```

[53]:
```
# Create a new DataFrame with only Dropout Rate and Graduate Rate
dropout_graduate_rates = student_courses_sorted[["Dropout Rate"]].copy().
 ↪round(3)

# Calculate the Graduate Rate
dropout_graduate_rates["Graduate Rate"] = 100 - dropout_graduate_rates[
    "Dropout Rate"
].round(3)

# Display the new DataFrame
dropout_graduate_rates
```

[53]:

Target	Dropout Rate	Graduate Rate
Course		
Biofuel Production Technologies	88.889	11.111
Oral Hygiene	47.826	52.174
Equinculture	65.000	35.000
Informatics Engineering	86.792	13.208
Basic Education	59.859	40.141
Communication Design	27.717	72.283
Agronomy	49.711	50.289
Animation and Multimedia Design	46.067	53.933
Social Service (evening attendance)	36.598	63.402
Tourism	45.498	54.502
Advertising and Marketing Management	43.182	56.818
Management (evening attendance)	63.551	36.449
Veterinary Nursing	34.351	65.649
Management	49.265	50.735
Journalism and Communication	34.007	65.993
Social Service	20.767	79.233
Nursing	17.718	82.282

0.0.1 Feature Selection

[55]:
```
df = pd.get_dummies(df, columns=["Target"])
df.head()
```

[55]:

	Marital status	Application mode	Application order	Course	\
0	1	8	5	2	
1	1	6	1	11	
2	1	1	5	5	
3	1	8	2	15	
4	2	12	1	3	

	Daytime/evening attendance	Previous qualification	Nationality	\
0	1	1	1	
1	1	1	1	
2	1	1	1	
3	1	1	1	
4	0	1	1	

	Mother's qualification	Father's qualification	Mother's occupation	…	\
0	13	10	6	…	
1	1	3	4	…	
2	22	27	10	…	
3	23	27	6	…	
4	22	28	10	…	

	Curricular units 2nd sem (enrolled)	\
0	0	
1	6	
2	6	
3	6	
4	6	

	Curricular units 2nd sem (evaluations)	\
0	0	
1	6	
2	0	
3	10	
4	6	

	Curricular units 2nd sem (approved)	Curricular units 2nd sem (grade)	\
0	0	0.000000	
1	6	13.666667	
2	0	0.000000	
3	5	12.400000	
4	6	13.000000	

	Curricular units 2nd sem (without evaluations)	Unemployment rate	\

```
0                                                 0              10.8
1                                                 0              13.9
2                                                 0              10.8
3                                                 0               9.4
4                                                 0              13.9

   Inflation rate   GDP  Target_Dropout  Target_Graduate
0             1.4  1.74            True            False
1            -0.3  0.79           False             True
2             1.4  1.74            True            False
3            -0.8 -3.12           False             True
4            -0.3  0.79           False             True

[5 rows x 36 columns]
```

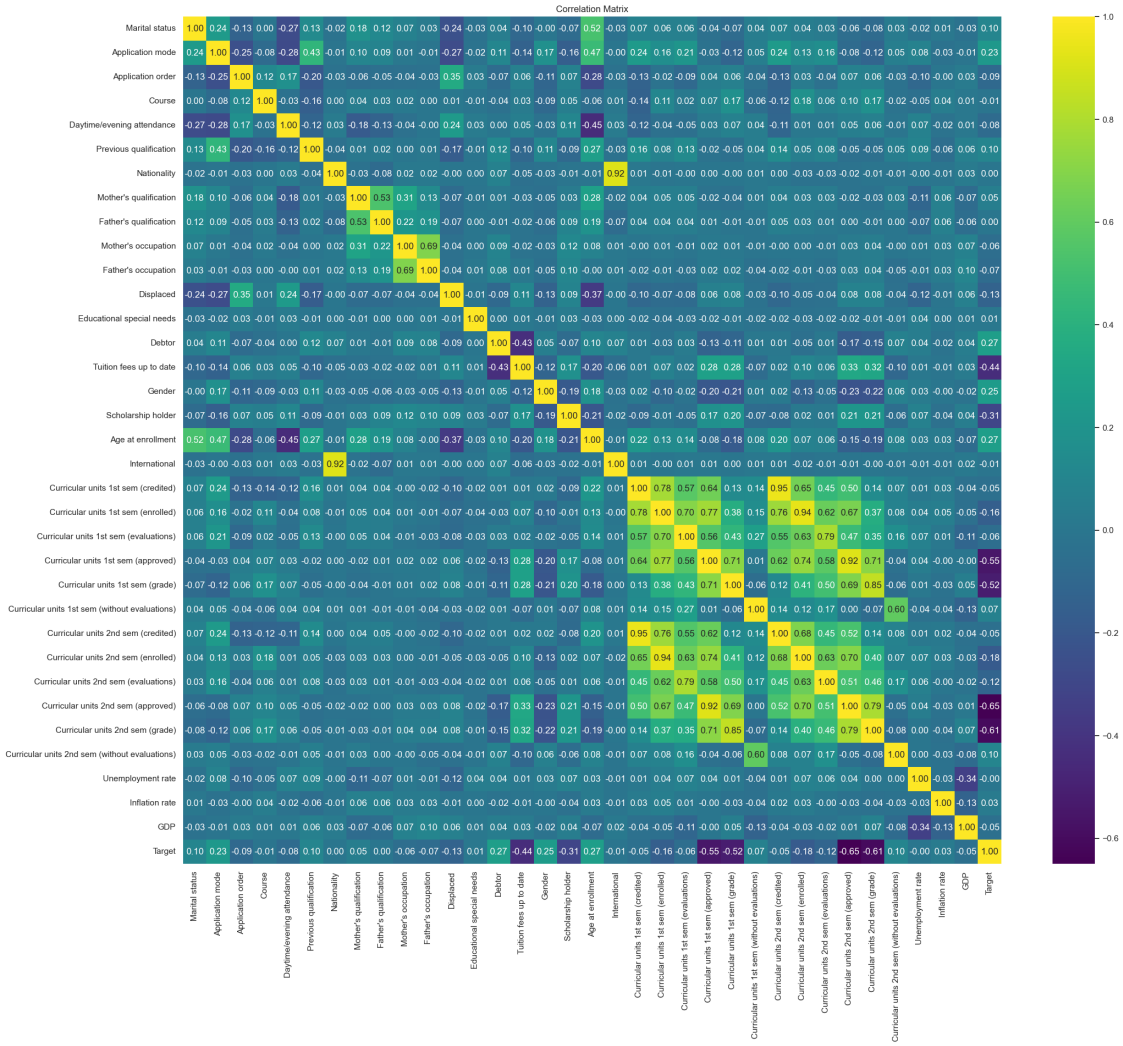
[56]:
```
dummies_to_drop = ["Target_Graduate"]
df.drop(columns=dummies_to_drop, inplace=True)
df.rename(columns={"Target_Dropout": "Target"}, inplace=True)

df.head()
```

[56]:
```
   Marital status  Application mode  Application order  Course  \
0               1                 8                  5       2
1               1                 6                  1      11
2               1                 1                  5       5
3               1                 8                  2      15
4               2                12                  1       3

   Daytime/evening attendance  Previous qualification  Nationality  \
0                           1                       1            1
1                           1                       1            1
2                           1                       1            1
3                           1                       1            1
4                           0                       1            1

   Mother's qualification  Father's qualification  Mother's occupation  …  \
0                      13                      10                    6  …
1                       1                       3                    4  …
2                      22                      27                   10  …
3                      23                      27                    6  …
4                      22                      28                   10  …

   Curricular units 2nd sem (credited)  Curricular units 2nd sem (enrolled)  \
0                                    0                                    0
1                                    0                                    6
2                                    0                                    6
3                                    0                                    6
```

```
4                                     0                                     6

   Curricular units 2nd sem (evaluations)  \
0                                       0
1                                       6
2                                       0
3                                      10
4                                       6

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)  \
0                                    0                          0.000000
1                                    6                         13.666667
2                                    0                          0.000000
3                                    5                         12.400000
4                                    6                         13.000000

   Curricular units 2nd sem (without evaluations)  Unemployment rate  \
0                                               0               10.8
1                                               0               13.9
2                                               0               10.8
3                                               0                9.4
4                                               0               13.9

   Inflation rate   GDP  Target
0             1.4  1.74    True
1            -0.3  0.79   False
2             1.4  1.74    True
3            -0.8 -3.12   False
4            -0.3  0.79   False

[5 rows x 35 columns]
```

[59]:
```python
# Set display options to show all columns and rows
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)

# Calculate the correlation matrix and round it to 2 decimal places
correlation_matrix = df.corr().round(2)

# Display the correlation matrix
correlation_matrix

# Reset display options to default values
pd.reset_option("display.max_columns")
pd.reset_option("display.max_rows")
```

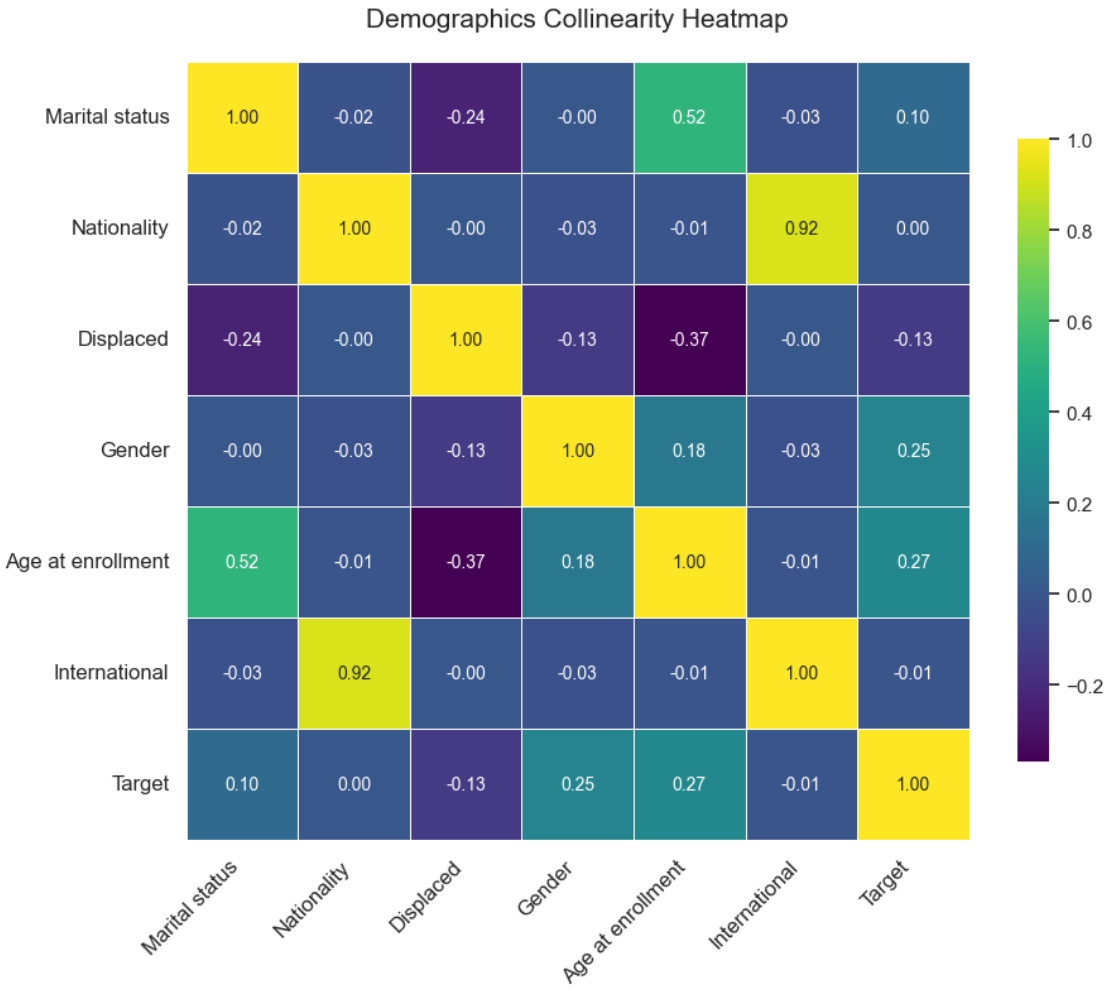
[67]:
```
sns.set(rc={"figure.figsize": (24, 20)})  # Increased the figure size
sns.heatmap(correlation_matrix, annot=True, cmap="viridis", fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



[64]:
```
# Demographic
demographics = df[
    [
        "Marital status",
        "Nationality",
        "Displaced",
        "Gender",
        "Age at enrollment",
        "International",
        "Target",
```

```
    ]
]
# Academic
academic_path = df[
    [
        "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)",
        "Target",
    ]
]
```

[68]:
```
sns.set(rc={"figure.figsize": (10, 8)})
sns.heatmap(
    demographics.corr().round(2),
    linewidths=0.5,
    annot=True,
    annot_kws={"size": 10},
    cmap="viridis",
    cbar_kws={"shrink": 0.8},
    fmt=".2f",
)

plt.title("Demographics Collinearity Heatmap", pad=20, fontsize=15)
plt.xticks(rotation=45, ha="right", fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



[70]:
```
features_to_drop = ["Nationality", "International"]
features_to_drop
```

[70]:
```
['Nationality', 'International']
```

[71]:
```
features_to_drop.extend(
    [
        "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 (without evaluations)",
    ]
```

```
)
features_to_drop
```

[71]:
```
['Nationality',
 '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 (without evaluations)']
```

[72]:
```
df.drop(features_to_drop, axis=1, inplace=True)
df.head()
```

[72]:

	Marital status	Application mode	Application order	Course	\
0	1	8	5	2	
1	1	6	1	11	
2	1	1	5	5	
3	1	8	2	15	
4	2	12	1	3	

	Daytime/evening attendance	Previous qualification	Mother's qualification	\
0	1	1	13	
1	1	1	1	
2	1	1	22	
3	1	1	23	
4	0	1	22	

	Father's qualification	Mother's occupation	Father's occupation	…	\
0	10	6	10	…	
1	3	4	4	…	
2	27	10	10	…	
3	27	6	4	…	
4	28	10	10	…	

	Scholarship holder	Age at enrollment	Curricular units 2nd sem (enrolled)	\
0	0	20	0	
1	0	19	6	
2	0	19	6	
3	0	20	6	
4	0	45	6	

	Curricular units 2nd sem (evaluations)	\
0	0	

```
1                                       6
2                                       0
3                                      10
4                                       6

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)  \
0                                    0                          0.000000
1                                    6                         13.666667
2                                    0                          0.000000
3                                    5                         12.400000
4                                    6                         13.000000

   Unemployment rate  Inflation rate   GDP  Target
0               10.8             1.4  1.74    True
1               13.9            -0.3  0.79   False
2               10.8             1.4  1.74    True
3                9.4            -0.8 -3.12   False
4               13.9            -0.3  0.79   False

[5 rows x 25 columns]
```

[73]:
```
df.corr()["Target"]
```

[73]:
```
Marital status                                0.100479
Application mode                              0.233888
Application order                            -0.094355
Course                                       -0.006814
Daytime/evening attendance                   -0.084496
Previous qualification                        0.102795
Mother's qualification                        0.048459
Father's qualification                        0.003850
Mother's occupation                          -0.064195
Father's occupation                          -0.073238
Displaced                                    -0.126113
Educational special needs                     0.007254
Debtor                                        0.267207
Tuition fees up to date                      -0.442138
Gender                                        0.251955
Scholarship holder                           -0.313018
Age at enrollment                             0.267229
Curricular units 2nd sem (enrolled)          -0.182897
Curricular units 2nd sem (evaluations)       -0.119239
Curricular units 2nd sem (approved)          -0.653995
Curricular units 2nd sem (grade)             -0.605350
Unemployment rate                            -0.004198
Inflation rate                                0.030326
GDP                                          -0.050260
```

```
Target                                          1.000000
Name: Target, dtype: float64
```

0.0.2 Logistic regression

[75]:
```
X = df.drop(columns="Target", axis=1)
y = df["Target"]
```

[76]:
```
X.shape
```

[76]:
```
(3630, 24)
```

[77]:
```
print("X: ", type(X))
print("y: ", type(y))
```

```
X:  <class 'pandas.core.frame.DataFrame'>
y:  <class 'pandas.core.series.Series'>
```

[ ]:
```
# Step 1: Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

# Step 2: Load and preprocess the data
# Assuming df is your DataFrame and 'Target' is the column you want to predict
# Replace 'feature1', 'feature2', ... with actual feature column names
X = df[["feature1", "feature2", "feature3"]]  # Features
y = df["Target"]  # Target variable

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Step 4: Train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Step 5: Evaluate the model
y_pred = model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Confusion matrix
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Classification report
class_report = classification_report(y_test, y_pred)
print("Classification Report:")
print(class_report)
```

[ ]: