In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('student_dropout.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	M quali
0	1	8	5	2	1	1	1	
1	1	6	1	11	1	1	1	
2	1	1	5	5	1	1	1	
3	1	8	2	15	1	1	1	
4	2	12	1	3	0	1	1	

5 rows × 35 columns

In [4]:

```
df.tail()
```

Out[4]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	q
4419	1	1	6	15	1	1	1	
4420	1	1	2	15	1	1	19	
4421	1	1	1	12	1	1	1	
4422	1	1	1	9	1	1	1	
4423	1	5	1	15	1	1	9	

5 rows × 35 columns

In [5]:

```
df.shape
```

Out[5]:

```
(4424, 35)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['Marital status', 'Application mode', 'Application order', 'Cours
e',
       'Daytime/evening attendance', 'Previous qualification', 'Nacionalit
y',
       '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', 'Internationa
l',
       '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 rat
e',
       'Inflation rate', 'GDP', 'Target'],
      dtype='object')
```

In [7]:

```
df.duplicated().sum()
```

Out[7]:

```
0
```

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
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
```

In [9]:

```
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   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   float
64
 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   float
64
 30  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 31  Unemployment rate                               4424 non-null   float
64
 32  Inflation rate                                  4424 non-null   float
64
 33  GDP                                             4424 non-null   float
64
 34  Target                                          4424 non-null   objec
t
dtypes: float64(5), int64(29), object(1)
memory usage: 1.2+ MB
```

In [10]:

```
df.describe()
```

Out[10]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4
mean	1.178571	6.886980	1.727848	9.899186	0.890823	2.531420	
std	0.605747	5.298964	1.313793	4.331792	0.311897	3.963707	
min	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	
25%	1.000000	1.000000	1.000000	6.000000	1.000000	1.000000	
50%	1.000000	8.000000	1.000000	10.000000	1.000000	1.000000	
75%	1.000000	12.000000	2.000000	13.000000	1.000000	1.000000	
max	6.000000	18.000000	9.000000	17.000000	1.000000	17.000000	

8 rows × 34 columns

In [11]:

```
df.nunique()
```

Out[11]:

```
Marital status                                   6
Application mode                                18
Application order                                8
Course                                          17
Daytime/evening attendance                       2
Previous qualification                          17
Nacionality                                     21
Mother's qualification                          29
Father's qualification                          34
Mother's occupation                             32
Father's occupation                             46
Displaced                                        2
Educational special needs                        2
Debtor                                           2
Tuition fees up to date                          2
Gender                                           2
Scholarship holder                               2
Age at enrollment                               46
International                                    2
Curricular units 1st sem (credited)             21
Curricular units 1st sem (enrolled)             23
Curricular units 1st sem (evaluations)          35
Curricular units 1st sem (approved)             23
Curricular units 1st sem (grade)               797
Curricular units 1st sem (without evaluations)  11
Curricular units 2nd sem (credited)             19
Curricular units 2nd sem (enrolled)             22
Curricular units 2nd sem (evaluations)          30
Curricular units 2nd sem (approved)             20
Curricular units 2nd sem (grade)               782
Curricular units 2nd sem (without evaluations)  10
Unemployment rate                               10
Inflation rate                                   9
GDP                                             10
Target                                           3
dtype: int64
```

df.rename(columns={'Marital status': 'Marital_status', 'Application mode': 'Application_mode', 'Application order': 'Application_order', 'Daytime/evening attendance': 'Daytime_evening_attendance', 'Previous qualification': 'Previous_qualification', "Mother's qualification"}, inplace=True)

In [12]:

```
df1 = df[['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',
          'Unemployment rate','Inflation rate', 'GDP', 'Target']]
```

In [13]:

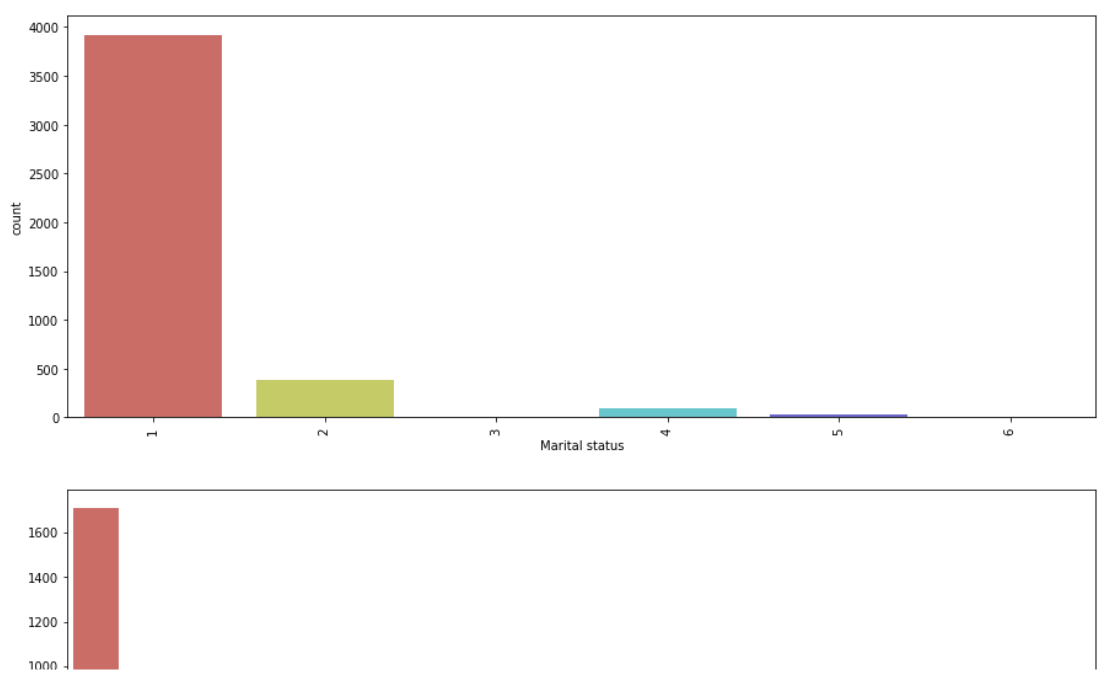
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:

```
import warnings
warnings.filterwarnings('ignore')
```

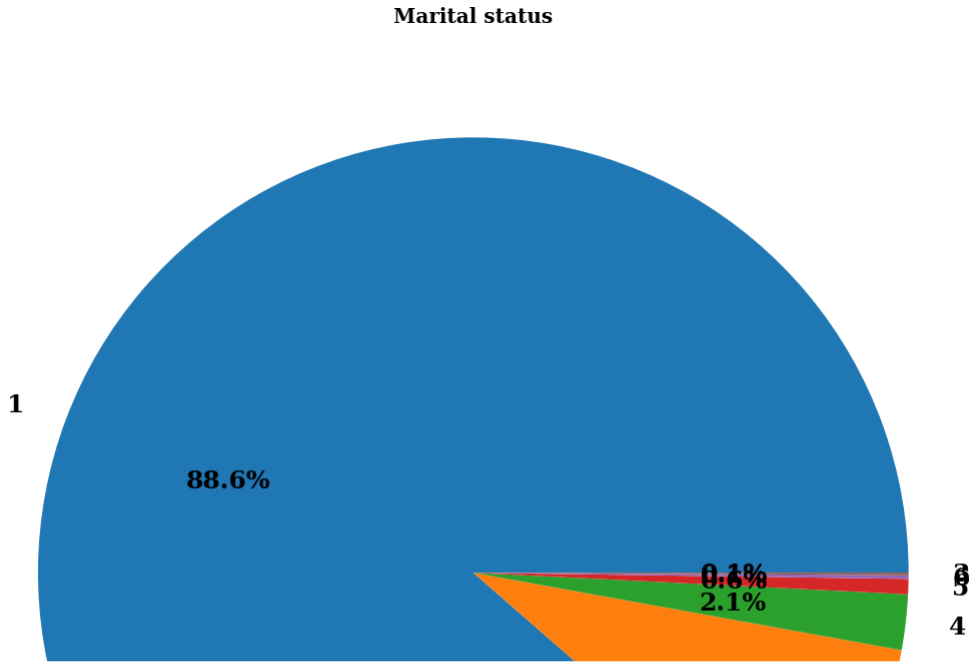
In [15]:

```
for i in df1.columns:
    plt.figure(figsize=(15,6))
    sns.countplot(df1[i], data = df, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



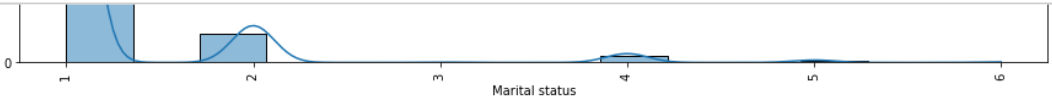
In [16]:

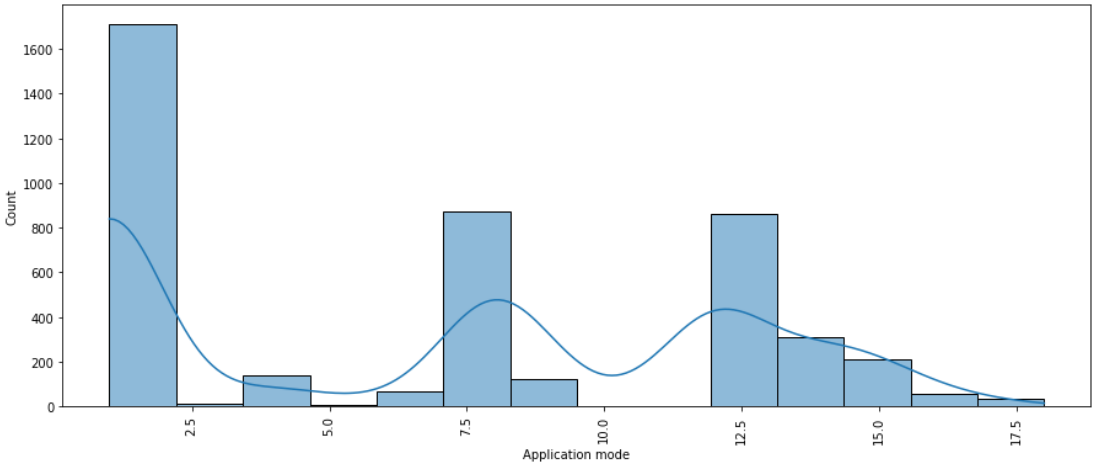
```
for i in df1.columns:
    plt.figure(figsize=(30,20))
    plt.pie(df1[i].value_counts(), labels=df1[i].value_counts().index, autopct='%1.1f%%'
                                          'color': 'black',
                                          'weight': 'bold',
                                          'family': 'serif' })
    hfont = {'fontname':'serif', 'weight': 'bold'}
    plt.title(i, size=20, **hfont)
    plt.show()
```



In [17]:

```
for i in df.columns:
    plt.figure(figsize=(15,6))
    sns.histplot(df[i], kde = True, palette = 'hls')
    plt.xticks(rotation = 90)
    plt.show()
```



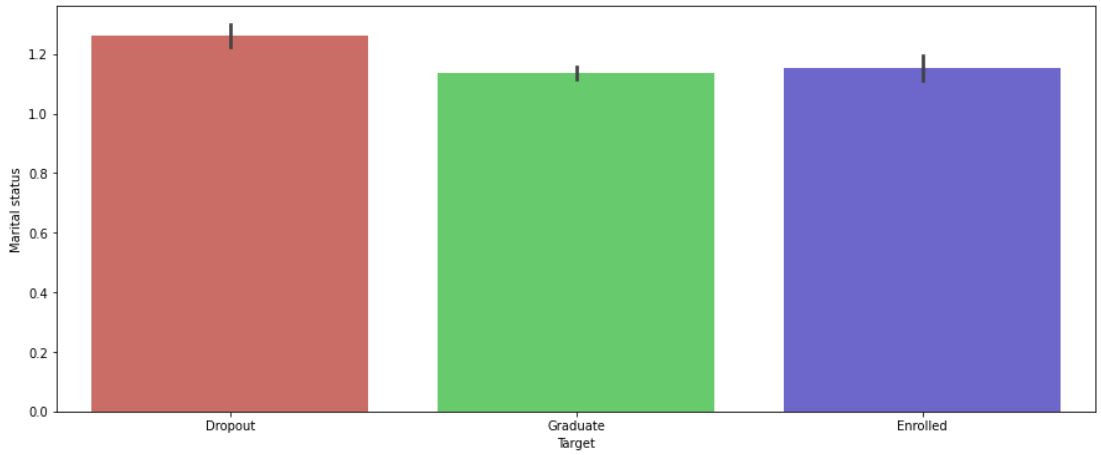


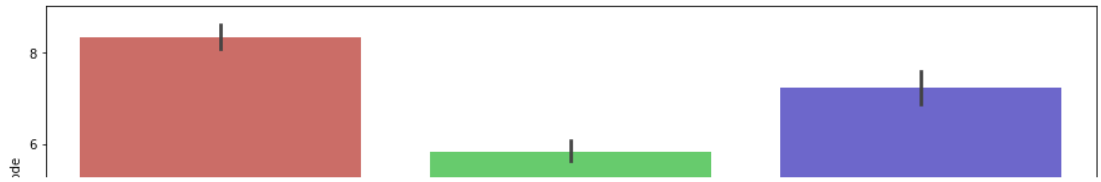
In [18]:

```
df2 = df.iloc[:, :-1]
```

In [19]:

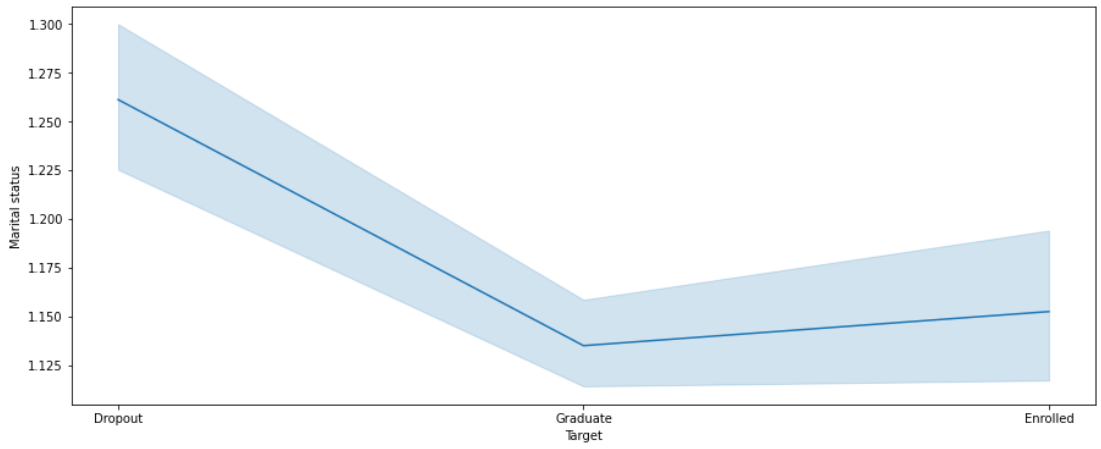
```
for i in df2.columns:
    plt.figure(figsize=(15,6))
    sns.barplot(x = df['Target'], y = df2[i], data = df, palette = 'hls')
    plt.show()
```





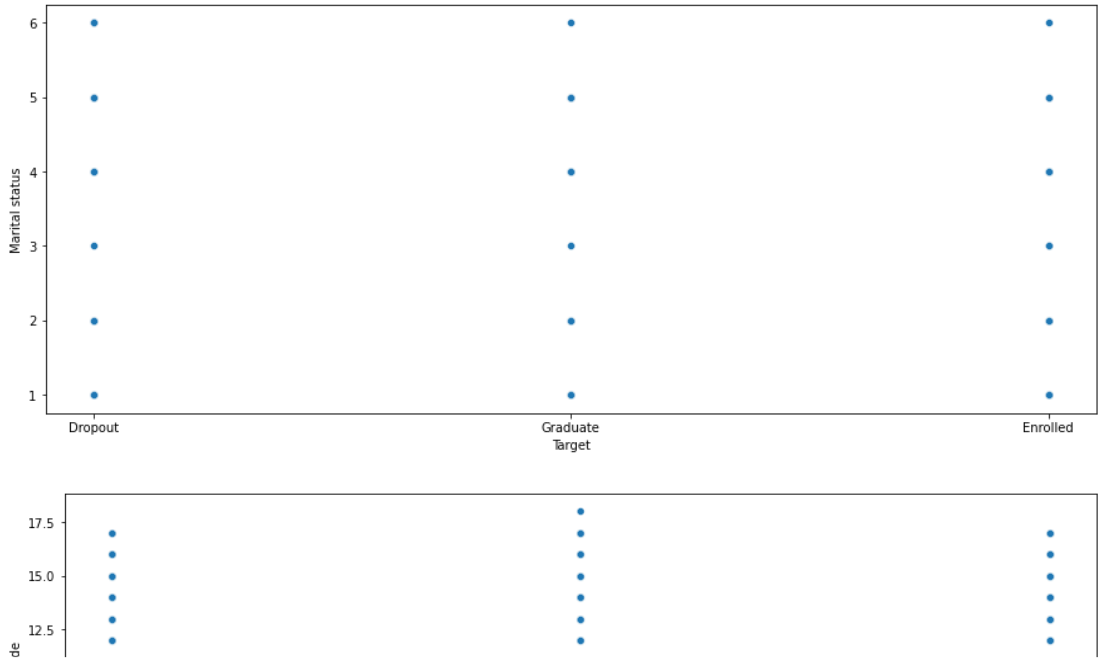
In [20]:

```
for i in df2.columns:
    plt.figure(figsize=(15,6))
    sns.lineplot(x = df['Target'], y = df2[i], data = df, palette = 'hls')
    plt.show()
```



In [21]:

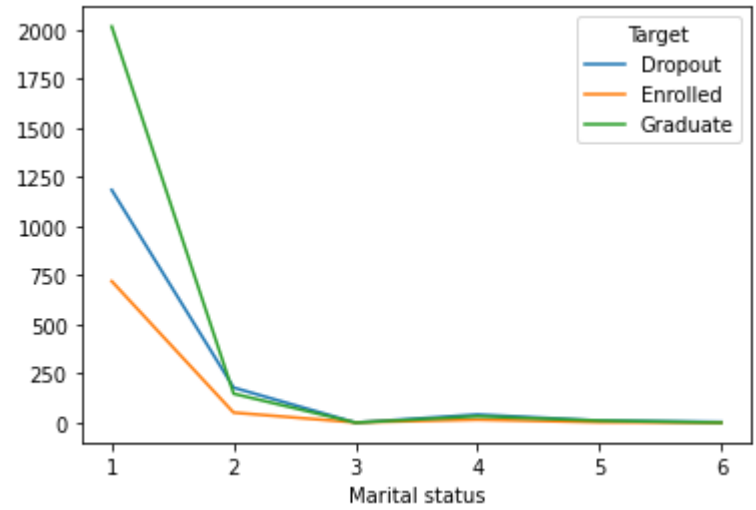
```
for i in df2.columns:
    plt.figure(figsize=(15,6))
    sns.scatterplot(x = df['Target'], y = df2[i], data = df, palette = 'hls')
    plt.show()
```



In [22]:

```
for i in df2.columns:
    plt.figure(figsize=(15,6))
    pd.crosstab(index=df2[i],columns=df['Target']).plot(kind='line')
    plt.show()
```

```
<Figure size 1080x432 with 0 Axes>
```




```
<Figure size 1080x432 with 0 Axes>
```

In [23]:

```
df_corr = df.corr()
```

In [24]:

```
df_corr
```

Out[24]:

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification
Marital status	1.000000	0.224855	-0.125854	0.018925	-0.274939	0.120925
Application mode	0.224855	1.000000	-0.246497	-0.085116	-0.268616	0.433028
Application order	-0.125854	-0.246497	1.000000	0.118928	0.158657	-0.199029
Course	0.018925	-0.085116	0.118928	1.000000	-0.070232	-0.158382
Daytime/evening attendance	-0.274939	-0.268616	0.158657	-0.070232	1.000000	-0.103022
Previous qualification	0.120925	0.433028	-0.199029	-0.158382	-0.103022	1.000000
Nacionality	-0.020722	-0.001360	-0.029385	-0.004761	0.024433	-0.038997
Mother's qualification	0.185522	0.092867	-0.061719	0.058909	-0.195346	0.018868
Father's qualification	0.128326	0.072798	-0.049936	0.045659	-0.137769	0.013152
Mother's occupation	0.069734	0.033489	-0.046591	0.029672	-0.037986	0.006190
Father's occupation	0.024351	0.001253	-0.029754	0.016489	0.000845	0.005381
Displaced	-0.234886	-0.263079	0.332362	0.006142	0.251767	-0.149356
Educational special needs	-0.028343	-0.030868	0.025597	-0.001886	0.031017	-0.015015
Debtor	0.034304	0.114348	-0.072151	-0.053149	0.006658	0.117447
Tuition fees up to date	-0.087158	-0.127339	0.055891	0.029099	0.038799	-0.095246
Gender	-0.014738	0.147226	-0.089559	-0.111383	-0.012326	0.089952
Scholarship holder	-0.053765	-0.152818	0.073709	0.051668	0.093912	-0.085668
Age at enrollment	0.522717	0.450700	-0.271154	-0.036929	-0.462280	0.249821
International	-0.027905	0.005050	-0.028801	-0.004662	0.027973	-0.033498
Curricular units 1st sem (credited)	0.061209	0.238269	-0.133354	-0.140546	-0.127466	0.159940
Curricular units 1st sem (enrolled)	0.052107	0.159547	-0.016808	0.112285	-0.043056	0.080860
Curricular units 1st sem (evaluations)	0.058030	0.219154	-0.092156	0.025970	-0.045889	0.129364
Curricular units 1st sem (approved)	-0.031027	-0.023713	0.035580	0.077038	0.016935	-0.005295

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification
Curricular units 1st sem (grade)	-0.059811	-0.106213	0.058308	0.179482	0.063974	-0.034252
Curricular units 1st sem (without evaluations)	0.034711	0.040255	-0.031699	-0.060483	0.045630	0.018276
Curricular units 2nd sem (credited)	0.062831	0.228973	-0.125815	-0.120390	-0.111953	0.138463
Curricular units 2nd sem (enrolled)	0.039026	0.127461	0.028878	0.185879	0.000371	0.056450
Curricular units 2nd sem (evaluations)	0.022784	0.164992	-0.055089	0.049236	0.014610	0.101501
Curricular units 2nd sem (approved)	-0.043739	-0.065203	0.071793	0.120000	0.034022	-0.037265
Curricular units 2nd sem (grade)	-0.071506	-0.104424	0.055517	0.178997	0.050493	-0.038765
Curricular units 2nd sem (without evaluations)	0.020426	0.042009	-0.015757	-0.013984	-0.004229	0.024186
Unemployment rate	-0.020338	0.091567	-0.098419	-0.050116	0.061974	0.096914
Inflation rate	0.008761	-0.019613	-0.011133	0.028775	-0.024043	-0.056388
GDP	-0.027003	-0.014563	0.030201	-0.012518	0.022929	0.053968

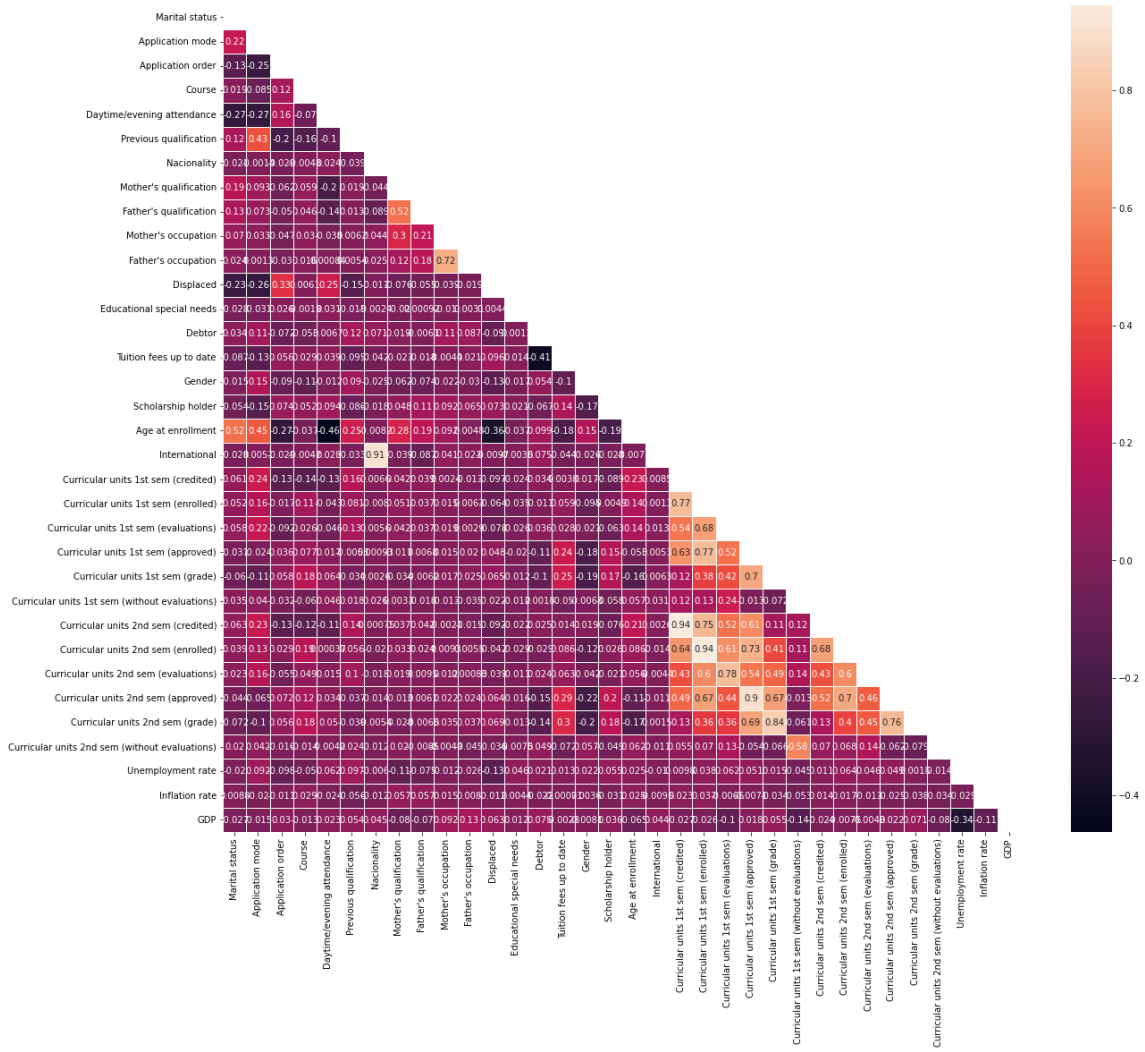
34 rows × 34 columns

In [25]:

```
import numpy as np
```

In [26]:

```
plt.figure(figsize=(20, 17))
matrix = np.triu(df_corr)
sns.heatmap(df_corr, annot=True, linewidth=.8, mask=matrix, cmap="rocket");
plt.show()
```



In [27]:

```
df['Target']=df['Target'].map({
    'Dropout':0,
    'Enrolled':1,
    'Graduate':2
})
```

In [28]:

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

In [29]:

```
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt
model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
```

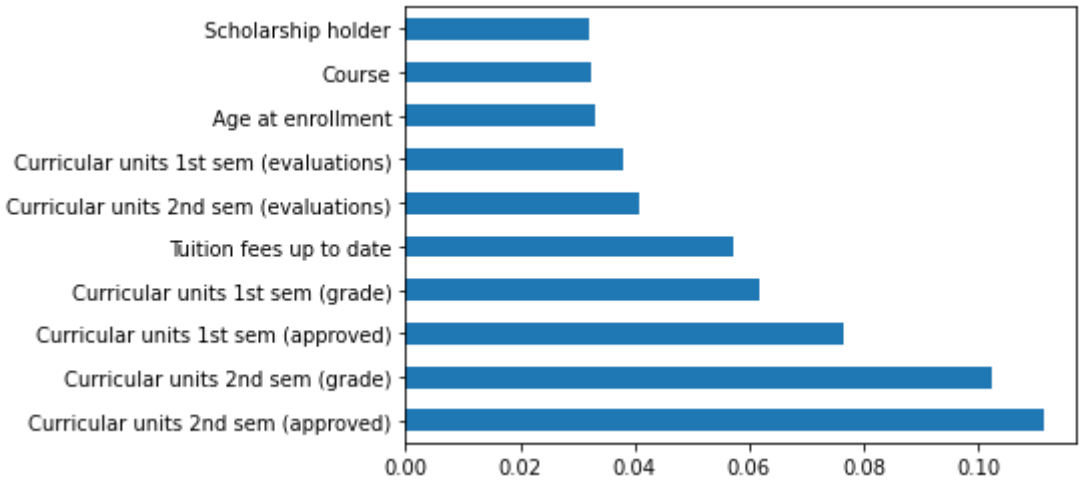
```
[0.00722764 0.02769016 0.02026333 0.03245958 0.00652237 0.00945437
 0.00335075 0.02811033 0.02712937 0.02973225 0.03182011 0.01616179
 0.0023769  0.01894743 0.05724495 0.0200354  0.03195265 0.03307028
 0.00302108 0.00950575 0.02550951 0.03800963 0.07636599 0.0618447
 0.00865291 0.00790895 0.02417979 0.040892   0.11159248 0.10255377
 0.00830515 0.02702742 0.02530893 0.0257723 ]
```

In [30]:

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

In [31]:

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



In [32]:

```
top_10 = pd.DataFrame({'Feature Importance': feat_importances.nlargest(10)})
```

In [33]:

```
top_10
```

Out[33]:

	Feature Importance
Curricular units 2nd sem (approved)	0.111592
Curricular units 2nd sem (grade)	0.102554
Curricular units 1st sem (approved)	0.076366
Curricular units 1st sem (grade)	0.061845
Tuition fees up to date	0.057245
Curricular units 2nd sem (evaluations)	0.040892
Curricular units 1st sem (evaluations)	0.038010
Age at enrollment	0.033070
Course	0.032460
Scholarship holder	0.031953

In [34]:

```
X = X[['Curricular units 2nd sem (approved)', 'Curricular units 2nd sem (grade)', 'Curri
       'Curricular units 1st sem (grade)', 'Tuition fees up to date', 'Curricular units 2
       'Age at enrollment', 'Course', 'Scholarship holder']]
```

In [35]:

```
from sklearn import preprocessing
```

In [36]:

```
scaler = preprocessing.MinMaxScaler()
X = scaler.fit_transform(X)
```

In [37]:

```
from sklearn.model_selection import train_test_split
```

In [38]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.25, random_state=
```

In [41]:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, con
```

In [39]:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(X_train, y_train)
```

Out[39]:

```
LogisticRegression
LogisticRegression()
```

In [40]:

```
y_pred = lr.predict(X_test)
```

In [44]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7350813743218807
```

In [45]:

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
```

Out[45]:

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [46]:

```
y_pred = dt.predict(X_test)
```

In [47]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6754068716094033
```

In [48]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

Out[48]:

```
RandomForestClassifier
RandomForestClassifier()
```

In [49]:

```
y_pred = rfc.predict(X_test)
```

In [50]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7486437613019892
```

In [51]:

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
```

In [52]:

```
folds = StratifiedKFold(n_splits = 5, shuffle = True, random_state = 40)
```

In [53]:

```
def grid_search(model,folds,params,scoring):
    grid_search = GridSearchCV(model,
                               cv=folds,
                               param_grid=params,
                               scoring=scoring,
                               n_jobs=-1, verbose=1)
    return grid_search
```

In [54]:

```
def print_best_score_params(model):
    print("Best Score: ", model.best_score_)
    print("Best Hyperparameters: ", model.best_params_)
```

In [55]:

```
log_reg = LogisticRegression()
log_params = {'C': [0.01, 1, 10],
          'penalty': ['l1', 'l2'],
          'solver': ['liblinear','newton-cg','saga']
         }
grid_search_log = grid_search(log_reg, folds, log_params, scoring=None)
grid_search_log.fit(X_train, y_train)
print_best_score_params(grid_search_log)
```

```
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best Score:  0.7522588089916226
Best Hyperparameters:  {'C': 10, 'penalty': 'l1', 'solver': 'saga'}
```

In [58]:

```
lr = LogisticRegression(C = 10, penalty = 'l1', solver = 'saga')
lr.fit(X_train, y_train)
```

Out[58]:

```
LogisticRegression(C=10, penalty='l1', solver='saga')
```

In [59]:

```
y_pred = lr.predict(X_test)
```

In [60]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.740506329113924
```

In [61]:

```
dtc = DecisionTreeClassifier(random_state=40)
dtc_params = {
    'max_depth': [5,10,20,30],
    'min_samples_leaf': [5,10,20,30]
}
grid_search_dtc = grid_search(dtc, folds, dtc_params, scoring='roc_auc_ovr')
grid_search_dtc.fit(X_train, y_train)
print_best_score_params(grid_search_dtc)
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best Score:  0.8534737881864685
Best Hyperparameters:  {'max_depth': 10, 'min_samples_leaf': 30}
```

In [62]:

```
dt = DecisionTreeClassifier(max_depth = 10, min_samples_leaf = 30)
dt.fit(X_train, y_train)
```

Out[62]:

```
DecisionTreeClassifier(max_depth=10, min_samples_leaf=30)
```

In [63]:

```
y_pred = dt.predict(X_test)
```

In [64]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7368896925858951
```

In [65]:

```
rfc = RandomForestClassifier(random_state=40, n_jobs = -1,oob_score=True)
rfc_params = {'max_depth': [10,20,30,40],
          'min_samples_leaf': [5,10,15,20,30],
          'n_estimators': [100,200,500,700]
        }
grid_search_rfc = grid_search(rfc, folds, rfc_params, scoring='roc_auc_ovr')
grid_search_rfc.fit(X_train, y_train)
print('OOB SCORE :',grid_search_rfc.best_estimator_.oob_score_)
```

```
Fitting 5 folds for each of 80 candidates, totalling 400 fits
OOB SCORE : 0.7613019891500904
```

In [66]:

```
print_best_score_params(grid_search_rfc)
```

```
Best Score:  0.879758909531772
Best Hyperparameters:  {'max_depth': 20, 'min_samples_leaf': 5, 'n_estimat
ors': 200}
```

In [67]:

```
rfc = RandomForestClassifier(max_depth = 20, min_samples_leaf = 5, n_estimators = 200)
rfc.fit(X_train, y_train)
```

Out[67]:

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=20
0)
```

In [68]:

```
y_pred = rfc.predict(X_test)
```

In [69]:

```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.759493670886076
```