Chaitanya Ambekar

TC77

In [4]:
```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import OneHotEncoder
```

In [5]:
```
df = pd.read_csv("Academic_Performance - Academic_Performance.csv")
df
```

Out[5]:

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM	COURSE 1 MARKS	COURSE 2 MARKS	COURSE 3 MARKS	COURSE 4 MARKS	COURSE 5 MARKS
0	SB11201210000129	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	71.0	93.0	71.0	93.0	79.0
1	SB11201210000137	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	97.0	38.0	86.0	98.0	78.0
2	SB11201210005154	M	No	Yes	ACADEMIC	ELECTRONIC ENGINEERING	17.0	1.0	18.0	43.0	22.0
3	SB11201210007504	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	65.0	35.0	76.0	80.0	48.0
4	SB11201210007548	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	94.0	94.0	98.0	100.0	71.0
...	...	...	...	...	...	...	...	...	...	...	...
12406	SB11201420568705	M	Yes	Yes	ACADEMIC	MECHATRONICS ENGINEERING	88.0	71.0	86.0	87.0	65.0
12407	SB11201420573045	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	46.0	39.0	44.0	11.0	0.0
12408	SB11201420578809	M	Yes	No	ACADEMIC	INDUSTRIAL ENGINEERING	98.0	88.0	90.0	81.0	87.0
12409	SB11201420578812	F	Yes	Yes	ACADEMIC	NaN	60.0	80.0	51.0	8.0	42.0
12410	SB11201420583232	M	No	No	ACADEMIC	INDUSTRIAL ENGINEERING	83.0	95.0	91.0	79.0	47.0

12411 rows × 13 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12411 entries, 0 to 12410
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   STUDENT_ID          12411 non-null  object 
 1   GENDER              12389 non-null  object 
 2   PLACEMENT           12396 non-null  object 
 3   HONOR_OPTED_OR_NOT  12397 non-null  object 
 4   EDUCATION_TYPE      12396 non-null  object 
 5   ACADEMIC_PROGRAM    12377 non-null  object 
 6   COURSE 1 MARKS      12400 non-null  float64
 7   COURSE 2 MARKS      12403 non-null  float64
 8   COURSE 3 MARKS      12397 non-null  float64
 9   COURSE 4 MARKS      12397 non-null  float64
 10  COURSE 5 MARKS      12389 non-null  float64
 11  PERCENTILE          12411 non-null  int64  
 12  OVEARLL_GRADE       12411 non-null  object 
dtypes: float64(5), int64(1), object(7)
memory usage: 1.2+ MB
```

In [4]:
```
df.describe()
```

Out[4]:

	COURSE 1 MARKS	COURSE 2 MARKS	COURSE 3 MARKS	COURSE 4 MARKS	COURSE 5 MARKS	PERCENTILE
count	12400.000000	12403.000000	12397.000000	12397.000000	12389.000000	12411.000000
mean	77.385887	62.191728	59.189562	67.501815	53.690532	68.446459
std	22.716227	27.669357	29.002446	25.505280	30.007993	25.867550
min	-1.000000	1.000000	1.000000	1.000000	0.000000	1.000000
25%	65.000000	42.000000	36.000000	51.000000	28.000000	51.000000
50%	85.000000	67.000000	65.000000	74.000000	56.000000	75.000000
75%	96.000000	86.000000	85.000000	88.000000	80.000000	90.000000
max	100.000000	100.000000	122.000000	111.000000	107.000000	100.000000

In [5]:
```
df.nunique()
```

Out[5]:
```
STUDENT_ID            12411
GENDER                    2
PLACEMENT                 2
HONOR_OPTED_OR_NOT        2
EDUCATION_TYPE            4
ACADEMIC_PROGRAM         21
COURSE 1 MARKS          101
COURSE 2 MARKS          100
COURSE 3 MARKS          105
COURSE 4 MARKS          102
COURSE 5 MARKS          102
PERCENTILE              100
OVEARLL_GRADE             4
dtype: int64
```

In [6]:
```
imputed_df = df
```

In [7]:
```
missing_values=imputed_df.isnull().sum()
print(missing_values)
```

```
STUDENT_ID             0
GENDER                22
PLACEMENT             15
HONOR_OPTED_OR_NOT    14
EDUCATION_TYPE        15
ACADEMIC_PROGRAM      34
COURSE 1 MARKS        11
COURSE 2 MARKS         8
COURSE 3 MARKS        14
COURSE 4 MARKS        14
COURSE 5 MARKS        22
PERCENTILE             0
OVEARLL_GRADE          0
dtype: int64
```

In [8]:
```
imputer = SimpleImputer(strategy='most_frequent', missing_values=np.nan)
imputed_df[['GENDER','PLACEMENT','EDUCATION_TYPE','ACADEMIC_PROGRAM']] = imputer.fit_transform(imputed_df[['GENDER','PLACEMENT','EDUCATION_
```

In [9]:
```
imputed_df['HONOR_OPTED_OR_NOT'] = imputed_df['HONOR_OPTED_OR_NOT'].fillna('No')
```

In [10]:
```
imputer1 = SimpleImputer(strategy='mean', missing_values=np.nan)
imputed_df[['COURSE 1 MARKS','COURSE 2 MARKS','COURSE 3 MARKS','COURSE 4 MARKS','COURSE 5 MARKS']] = imputer1.fit_transform(imputed_df[['CO
```

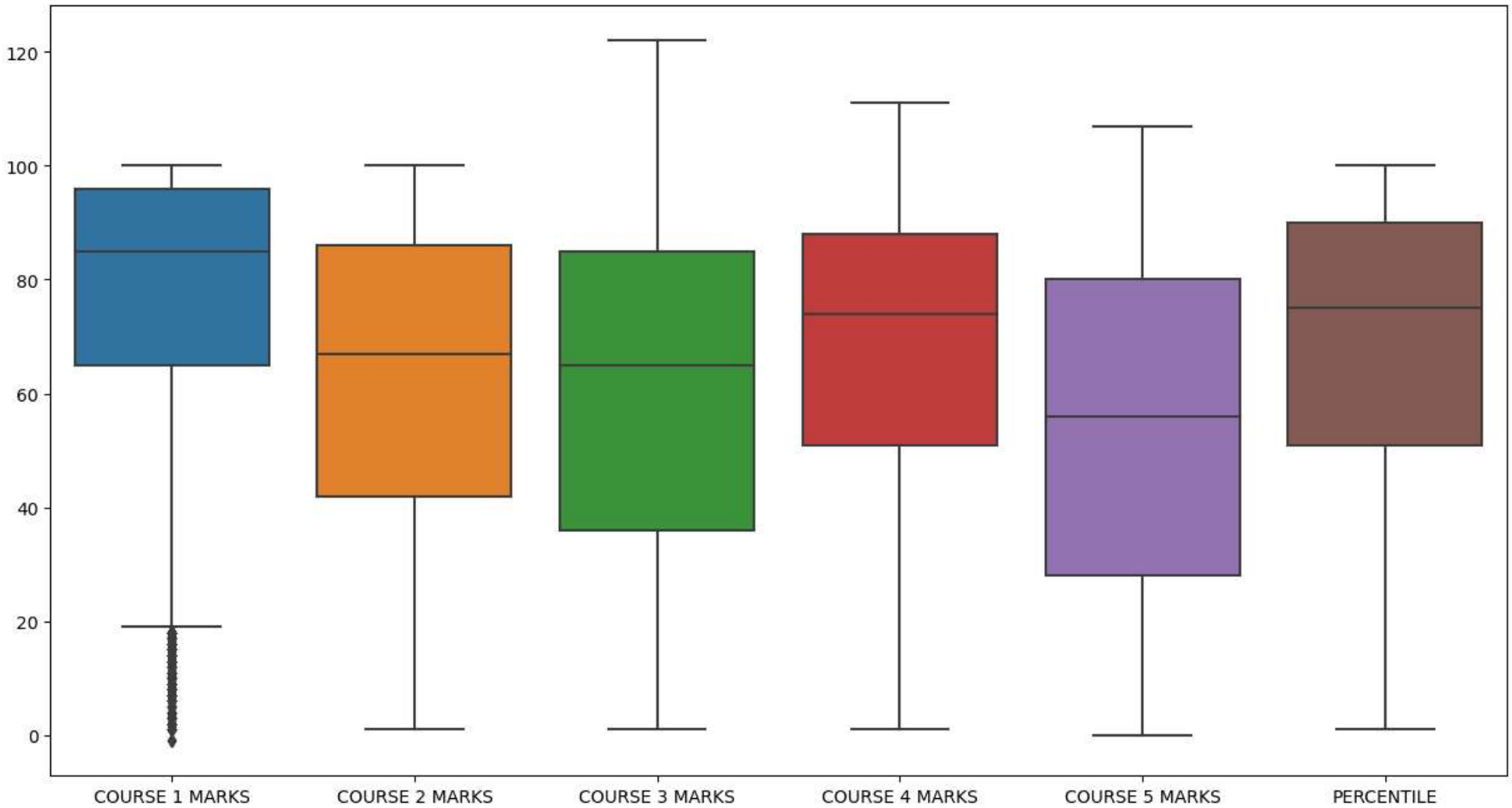
In [11]:
```
missing_values=imputed_df.isnull().sum()
print(missing_values)
```

```
STUDENT_ID            0
GENDER                0
PLACEMENT             0
HONOR_OPTED_OR_NOT    0
EDUCATION_TYPE        0
ACADEMIC_PROGRAM      0
COURSE 1 MARKS        0
COURSE 2 MARKS        0
COURSE 3 MARKS        0
COURSE 4 MARKS        0
COURSE 5 MARKS        0
PERCENTILE            0
OVEARLL_GRADE         0
dtype: int64
```

Detecting outliers

In [12]:
```
plt.figure(figsize=(15,8))
sns.boxplot(data=imputed_df)
```

Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
df1 = imputed_df
```

In [14]:
```
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers
marks_outliers = detect_outliers_iqr(df1['COURSE 1 MARKS'])
print("Outliers from IQR method: ", marks_outliers)
print("\nTotal number of outliers = ",np.count_nonzero(marks_outliers))
```

```
Outliers from IQR method:  [-1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0,
2.0, 2.0, 2.0, 2.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 4.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0,
6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 6.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0, 7.0,
8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 8.0, 9.0, 9.0, 9.0, 9.0, 9.0, 9.0, 9.0, 9.0,
9.0, 9.0, 9.0, 9.0, 9.0, 9.0, 9.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 1
0.0, 10.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 11.0, 12.0, 12.0, 12.0, 12.0, 12.0, 1
2.0, 12.0, 12.0, 12.0, 12.0, 12.0, 12.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 13.0, 1
3.0, 13.0, 13.0, 13.0, 13.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 14.0, 15.0, 1
5.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 15.0, 1
6.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 16.0, 17.0, 17.0, 1
7.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 17.0, 1
8.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 18.0, 1
8.0, 18.0, 18.0]

Total number of outliers =  294
```

In [15]:
```
Q1=df1['COURSE 1 MARKS'].quantile(0.25)
Q3=df1['COURSE 1 MARKS'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1-1.5*IQR
Upper_Whisker = Q3+1.5*IQR
print(Lower_Whisker, Upper_Whisker)
```

```
65.0
96.0
31.0
18.5 142.5
```

In [16]:
```
df1.describe()
```

Out[16]:

	COURSE 1 MARKS	COURSE 2 MARKS	COURSE 3 MARKS	COURSE 4 MARKS	COURSE 5 MARKS	PERCENTILE
count	12411.000000	12411.000000	12411.000000	12411.000000	12411.000000	12411.000000
mean	77.385887	62.191728	59.189562	67.501815	53.690532	68.446459
std	22.706157	27.660437	28.986082	25.490889	29.981382	25.867550
min	-1.000000	1.000000	1.000000	1.000000	0.000000	1.000000
25%	65.000000	42.000000	36.000000	51.000000	28.000000	51.000000
50%	85.000000	67.000000	65.000000	74.000000	56.000000	75.000000
75%	96.000000	86.000000	85.000000	88.000000	80.000000	90.000000
max	100.000000	100.000000	122.000000	111.000000	107.000000	100.000000

In [17]:
```
#Apply conditions to remove outliers
df1 = df1[df1['COURSE 1 MARKS'] < Upper_Whisker]
df1 = df1[df1['COURSE 1 MARKS'] > Lower_Whisker]
```

In [18]:
```
df1.describe()
```

Out[18]:

	COURSE 1 MARKS	COURSE 2 MARKS	COURSE 3 MARKS	COURSE 4 MARKS	COURSE 5 MARKS	PERCENTILE
count	12117.000000	12117.000000	12117.000000	12117.000000	12117.000000	12117.000000
mean	78.998370	63.093714	59.985777	68.255676	54.068102	69.641660
std	20.436679	27.184343	28.696120	25.043541	29.921293	24.862564
min	19.000000	1.000000	1.000000	1.000000	0.000000	1.000000
25%	67.000000	43.000000	37.000000	52.000000	28.000000	53.000000
50%	86.000000	69.000000	65.000000	75.000000	56.000000	76.000000
75%	96.000000	86.000000	85.000000	88.000000	81.000000	91.000000
max	100.000000	100.000000	122.000000	111.000000	107.000000	100.000000

Label and OneHot Encoding

In [19]:
```
df_cat = df1.select_dtypes(exclude=[np.number])
df_cat
```

Out[19]:

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM	OVEARLL_GRADE
0	SB11201210000129	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
1	SB11201210000137	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	THIRD CLASS
3	SB11201210007504	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
4	SB11201210007548	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
5	SB11201210007568	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
...	...	...	...	...	...	...	...
12406	SB11201420568705	M	Yes	Yes	ACADEMIC	MECHATRONICS ENGINEERING	FIRST CLASS
12407	SB11201420573045	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
12408	SB11201420578809	M	Yes	No	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
12409	SB11201420578812	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLASS
12410	SB11201420583232	M	No	No	ACADEMIC	INDUSTRIAL ENGINEERING	THIRD CLASS

12117 rows × 7 columns

In [20]:
```
df_cat.nunique()
```

Out[20]:
```
STUDENT_ID            12117
GENDER                    2
PLACEMENT                 2
HONOR_OPTED_OR_NOT        2
EDUCATION_TYPE            4
ACADEMIC_PROGRAM         21
OVEARLL_GRADE             4
dtype: int64
```

In [21]:
```
cols = ['GENDER','PLACEMENT','HONOR_OPTED_OR_NOT','EDUCATION_TYPE','OVEARLL_GRADE']
le = LabelEncoder()
df_cat[cols] = df_cat[cols].apply(le.fit_transform)
#df_cat[cols] = le.fit_transform(df_cat[cols])
```

In [22]:
```
df_cat
```

Out[22]:

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM	OVEARLL_GRADE
0	SB11201210000129	0	1	1	0	INDUSTRIAL ENGINEERING	1
1	SB11201210000137	0	1	1	0	INDUSTRIAL ENGINEERING	3
3	SB11201210007504	0	1	1	0	INDUSTRIAL ENGINEERING	1
4	SB11201210007548	1	1	1	0	INDUSTRIAL ENGINEERING	1
5	SB11201210007568	0	1	1	0	INDUSTRIAL ENGINEERING	1
...	...	...	...	...	...	...	...
12406	SB11201420568705	1	1	1	0	MECHATRONICS ENGINEERING	1
12407	SB11201420573045	1	1	1	0	INDUSTRIAL ENGINEERING	1
12408	SB11201420578809	1	1	0	0	INDUSTRIAL ENGINEERING	1
12409	SB11201420578812	0	1	1	0	INDUSTRIAL ENGINEERING	1
12410	SB11201420583232	1	0	0	0	INDUSTRIAL ENGINEERING	3

12117 rows × 7 columns

In [23]:
```
df_cat = df_cat.iloc[:,1:]
del df_cat['ACADEMIC_PROGRAM']
df_cat
```

Out[23]:

	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	OVEARLL_GRADE
0	0	1	1	0	1
1	0	1	1	0	3
3	0	1	1	0	1
4	1	1	1	0	1
5	0	1	1	0	1
...	...	...	...	...	...
12406	1	1	1	0	1
12407	1	1	1	0	1
12408	1	1	0	0	1
12409	0	1	1	0	1
12410	1	0	0	0	3

12117 rows × 5 columns

In [24]:
```
cols = ['EDUCATION_TYPE','OVEARLL_GRADE']
onehot_encoder = OneHotEncoder(sparse=False)

df_one = onehot_encoder.fit_transform(df_cat[cols])
```

In [25]:
```
onehot_encoded_frame = pd.DataFrame(df_one, columns = onehot_encoder.get_feature_names(cols))
onehot_encoded_frame
```

```
D:\juypter\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[25]:

	EDUCATION_TYPE_0	EDUCATION_TYPE_1	EDUCATION_TYPE_2	EDUCATION_TYPE_3	OVEARLL_GRADE_0	OVEARLL_GRADE_1	OVEARLL_GRADE_2	OVEARLL_GRA
0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
3	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
4	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
...	...	...	...	...	...	...	...	
12112	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
12113	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
12114	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
12115	1.0	0.0	0.0	0.0	0.0	1.0	0.0	
12116	1.0	0.0	0.0	0.0	0.0	0.0	0.0	

12117 rows × 8 columns

In [ ]: