Analyzing The Combined Dataset

First let's import the necessary libraries.

In [223…
```
import numpy as np
import pandas as pd
import os
import random
import scipy.stats as st

random.seed(42)
```

Also import the visualization libraries.

In [224…
```
%matplotlib inline

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

plt.style.use('ggplot')
```

Let's define a function so that we can easily load the datasets.

In [225…
```
DATASET_PATH = 'Workable Datasets'

def load_the_dataset(file_name, dataset_path=DATASET_PATH):
    csv_path = os.path.join(dataset_path, file_name)
    return pd.read_csv(csv_path)
```

Let's import the dataset.

In [226…
```
university_df = load_the_dataset('UNIVERSITY_N.csv')
college_df = load_the_dataset('COLLEGE_N.csv')
school_df =  load_the_dataset('SCHOOL_N.csv')
```

Let's check the data.

In [227…
```
university_df.head()
```

Out[227…

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Resider
0	Female	23	Instagram	Not at all	Desktop	Library	Connected	Night	Daily	Remo
1	Female	23	Youtube	Good	Mobile	University	Connected	Morning	Daily	Remo

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Resider
2	Female	23	Whatsapp	Good	Mobile	University	Connected	Midnight	Daily	To
3	Female	23	Whatsapp	Average	Laptop and Mobile	University	Connected	Morning	Daily	Villa

In [228…
```
college_df.head()
```

Out[228…

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Residence
0	Female	17	Google	Very Good	Mobile	Home	Not Connected	Night	Daily	Town
1	Female	17	Facebook	Good	Mobile	Home	Not Connected	Night	Daily	Town
2	Female	17	Youtube	Very Good	Mobile	Home	Not Connected	Night	Daily	Town
3	Female	18	Youtube	Good	Mobile	Home	Not Connected	Night	Weekly	Town
4	Male	17	Whatsapp	Very Good	Mobile	Home	Not Connected	Night	Daily	Town

In [229…
```
school_df.head()
```

Out[229…

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Residenc

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Residenc
0	Male	15	Google	Very Good	Mobile	Home	Not Connected	Night	Daily	Tow
1	Female	14	Google	Very Good	Mobile	Home	Not Connected	Night	Daily	Villag
2	Male	16	Facebook	Not at all	Mobile	Home	Not Connected	Night	Weekly	Tow
3	Male	14	Facebook	Very Good	Mobile	Home	Not Connected	Morning	Daily	Tow

Check the dataset using `info()` .

In [230…
```
university_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 20 columns):
 #   Column                                  Non-Null Count  Dtype 
---  ------                                  --------------  ----- 
 0   Gender                                  301 non-null    object
 1   Age                                     301 non-null    int64 
 2   Popular Website                         301 non-null    object
 3   Proficiency                             301 non-null    object
 4   Medium                                  301 non-null    object
 5   Location                                301 non-null    object
 6   Household Internet Facilities           301 non-null    object
 7   Browse Time                             301 non-null    object
 8   Browsing Status                         301 non-null    object
 9   Residence                               301 non-null    object
 10  Total Internet Usage(hrs/day)           301 non-null    int64 
 11  Time Spent in Academic(hrs/day)         301 non-null    int64 
 12  Purpose of Use                          301 non-null    object
 13  Years of Internet Use                   301 non-null    int64 
 14  Browsing Purpose                        301 non-null    object
 15  Webinar                                 301 non-null    object
 16  Priority of Learning                    301 non-null    object
 17  Internet Usage For Educational Purpose  301 non-null    object
 18  Academic Performance                    301 non-null    object
 19  Obstacles                               301 non-null    object
dtypes: int64(4), object(16)
memory usage: 47.2+ KB
```

In [231…
```
college_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
```

```
Data columns (total 20 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Gender                                  199 non-null    object
 1   Age                                     199 non-null    int64
 2   Popular Website                         199 non-null    object
 3   Proficiency                             199 non-null    object
 4   Medium                                  199 non-null    object
 5   Location                                199 non-null    object
 6   Household Internet Facilities           199 non-null    object
 7   Browse Time                             199 non-null    object
 8   Browsing Status                         199 non-null    object
 9   Residence                               199 non-null    object
 10  Total Internet Usage(hrs/day)           199 non-null    int64
 11  Time Spent in Academic(hrs/day)         199 non-null    int64
 12  Purpose of Use                          199 non-null    object
 13  Years of Internet Use                   199 non-null    int64
 14  Browsing Purpose                        199 non-null    object
 15  Priority of Learning                    199 non-null    object
 16  Webinar                                 199 non-null    object
 17  Internet Usage For Educational Purpose  199 non-null    object
 18  Academic Performance                    199 non-null    object
 19  Obstacles                               199 non-null    object
dtypes: int64(4), object(16)
memory usage: 31.2+ KB
```

In [232…
```
school_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 20 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Gender                                  199 non-null    object
 1   Age                                     199 non-null    int64
 2   Popular Website                         199 non-null    object
 3   Proficiency                             199 non-null    object
 4   Medium                                  199 non-null    object
 5   Location                                199 non-null    object
 6   Household Internet Facilities           199 non-null    object
 7   Browse Time                             199 non-null    object
 8   Browsing Status                         199 non-null    object
 9   Residence                               199 non-null    object
 10  Total Internet Usage(hrs/day)           199 non-null    int64
 11  Time Spent in Academic(hrs/day)         199 non-null    int64
 12  Purpose of Use                          199 non-null    object
 13  Years of Internet Use                   199 non-null    int64
 14  Browsing Purpose                        199 non-null    object
 15  Priority of Learning                    199 non-null    object
 16  Webinar                                 199 non-null    object
 17  Internet Usage For Educational Purpose  199 non-null    object
 18  Academic Performance                    199 non-null    object
 19  Obstacles                               199 non-null    object
dtypes: int64(4), object(16)
memory usage: 31.2+ KB
```

Let's check the shape .

In [233…
```
university_df.shape
```

Out[233… (301, 20)

In [234…
```
college_df.shape
```

Out[234… (199, 20)

In [235…
```
school_df.shape
```

Out[235… (199, 20)

Combine The Datasets

We'll now combine the datasets together. But first let's add a column called `Academic Institution` designating which academic institution each student belongs to.

Start with the `university_df` .

In [236…
```
new_list_u = []

for value in university_df['Age']:
    new_list_u.append('University')
```

In [237…
```
university_df.insert(2, 'Academic Institution', new_list_u, True)
```

Let's check the data.

In [238…
```
university_df.head()
```

Out[238…

	Gender	Age	Academic Institution	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Brows Sta
0	Female	23	University	Instagram	Not at all	Desktop	Library	Connected	Night	D
1	Female	23	University	Youtube	Good	Mobile	University	Connected	Morning	D
2	Female	23	University	Whatsapp	Good	Mobile	University	Connected	Midnight	D
3	Female	23	University	Whatsapp	Average	Laptop and Mobile	University	Connected	Morning	D
4	Male	24	University	Facebook	Average	Laptop and Mobile	Cyber Cafe	Connected	Night	D

5 rows × 21 columns

Let's check the values in `Academic Institution` .

In [239…
```
university_df['Academic Institution'].value_counts()
```

Out[239…
```
University    301
Name: Academic Institution, dtype: int64
```

Now do the same operation to `college_df` .

In [240…
```
new_list_c = []

for value in college_df['Age']:
    new_list_c.append('College')
```

In [241…
```
college_df.insert(2, 'Academic Institution', new_list_c, True)
```

Let's check the data.

In [242…
```
college_df.head()
```

Out[242…

	Gender	Age	Academic Institution	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsin Statu
0	Female	17	College	Google	Very Good	Mobile	Home	Not Connected	Night	Dail
1	Female	17	College	Facebook	Good	Mobile	Home	Not Connected	Night	Dail
2	Female	17	College	Youtube	Very Good	Mobile	Home	Not Connected	Night	Dail
3	Female	18	College	Youtube	Good	Mobile	Home	Not Connected	Night	Weekl
4	Male	17	College	Whatsapp	Very Good	Mobile	Home	Not Connected	Night	Dail

5 rows × 21 columns

Let's check the values in `Academic Institution` .

In [243…
```
college_df['Academic Institution'].value_counts()
```

Out[243…
```
College    199
Name: Academic Institution, dtype: int64
```

Now do the same operation to `school_df` .

In [244…
```
new_list_s = []

for value in school_df['Age']:
    new_list_s.append('School')
```

In [245…
```
school_df.insert(2, 'Academic Institution', new_list_s, True)
```

Let's check the data.

In [246…
```
school_df.head()
```

Out[246…

	Gender	Age	Academic Institution	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsi Stat
0	Male	15	School	Google	Very Good	Mobile	Home	Not Connected	Night	Da
1	Female	14	School	Google	Very Good	Mobile	Home	Not Connected	Night	Da
2	Male	16	School	Facebook	Not at all	Mobile	Home	Not Connected	Night	Week
3	Male	14	School	Facebook	Very Good	Mobile	Home	Not Connected	Morning	Da
4	Female	14	School	Whatsapp	Very Good	Mobile	Home	Not Connected	Night	Da

5 rows × 21 columns

Let's check the values in `Academic Institution` .

In [247…
```
school_df['Academic Institution'].value_counts()
```

Out[247…
```
School    199
Name: Academic Institution, dtype: int64
```

Finally, let's combine datasets.

In [248…
```
frames = [university_df, college_df, school_df]

combined_df = pd.concat(frames)
```

Let's check the data.

In [249…
```
combined_df
```

Out[249…

	Gender	Age	Academic Institution	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Bro
0	Female	23	University	Instagram	Not at all	Desktop	Library	Connected	Night	
1	Female	23	University	Youtube	Good	Mobile	University	Connected	Morning	
2	Female	23	University	Whatsapp	Good	Mobile	University	Connected	Midnight	
3	Female	23	University	Whatsapp	Average	Laptop and Mobile	University	Connected	Morning	
4	Male	24	University	Facebook	Average	Laptop and Mobile	Cyber Cafe	Connected	Night	
...	...	...	...	...	...	...	...	...	...	
194	Male	14	School	Facebook	Very Good	Mobile	Home	Not Connected	Night	
195	Female	14	School	Google	Very Good	Mobile	Home	Not Connected	Night	
196	Male	15	School	Facebook	Very Good	Mobile	Home	Not Connected	Night	W

	Gender	Age	Academic Institution	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Bro
197	Female	15	School	Youtube	Average	Laptop and Mobile	Home	Not Connected	Night	W
198	Male	15	School	Facebook	Average	Laptop and	Home	Not	Night	

Let's check the `shape` of the new dataset.

In [250…
```
combined_df.shape
```

Out[250…
```
(699, 21)
```

In [251…
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 0 to 198
Data columns (total 21 columns):
 #   Column                                 Non-Null Count  Dtype 
---  ------                                 --------------  ----- 
 0   Gender                                 699 non-null    object
 1   Age                                    699 non-null    int64 
 2   Academic Institution                   699 non-null    object
 3   Popular Website                        699 non-null    object
 4   Proficiency                            699 non-null    object
 5   Medium                                 699 non-null    object
 6   Location                               699 non-null    object
 7   Household Internet Facilities          699 non-null    object
 8   Browse Time                            699 non-null    object
 9   Browsing Status                        699 non-null    object
 10  Residence                              699 non-null    object
 11  Total Internet Usage(hrs/day)          699 non-null    int64 
 12  Time Spent in Academic(hrs/day)        699 non-null    int64 
 13  Purpose of Use                         699 non-null    object
 14  Years of Internet Use                  699 non-null    int64 
 15  Browsing Purpose                       699 non-null    object
 16  Webinar                                699 non-null    object
 17  Priority of Learning                   699 non-null    object
 18  Internet Usage For Educational Purpose 699 non-null    object
 19  Academic Performance                   699 non-null    object
 20  Obstacles                              699 non-null    object
dtypes: int64(4), object(17)
memory usage: 120.1+ KB
```

Now let's check all the categorical attributes individually. Start with `Gender` first.

In [252…
```
combined_df['Gender'].value_counts()
```

Out[252…
```
Male      368
Female    331
```

Check Age

In [253…
```
combined_df['Age'].value_counts()
```

Out[253…
```
23    189
17    180
14    103
15     90
24     76
25     30
16     13
18     12
22      4
26      1
20      1
Name: Age, dtype: int64
```

Check Academic Institution

In [254…
```
combined_df['Academic Institution'].value_counts()
```

Out[254…
```
University    301
College       199
School        199
Name: Academic Institution, dtype: int64
```

Check Frequently Visited Website

In [255…
```
combined_df['Popular Website'].value_counts()
```

Out[255…
```
Google       247
Youtube      147
Facebook     113
Whatsapp     108
Gmail         43
Twitter       24
Instagram     17
Name: Popular Website, dtype: int64
```

In [256…
```
combined_df.rename(columns={
    'Popular Website':'Frequently Visited Website',
}, inplace=True)

combined_df.columns
```

Out[256…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Proficiency', 'Medium', 'Location', 'Household Internet Facilities',
       'Browse Time', 'Browsing Status', 'Residence',
       'Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
       'Purpose of Use', 'Years of Internet Use', 'Browsing Purpose',
       'Webinar', 'Priority of Learning',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Effectiveness Of Internet Usage

In [257…
```
combined_df['Proficiency'].value_counts()
```

Out[257…
```
Very Good     280
Good          196
Average       186
Not at all     37
Name: Proficiency, dtype: int64
```

In [258…
```
combined_df.rename(columns={
    'Proficiency':'Effectiveness Of Internet Usage'
}, inplace=True)

combined_df.columns
```

Out[258…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Medium', 'Location',
       'Household Internet Facilities', 'Browse Time', 'Browsing Status',
       'Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose of Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority of Learning', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

In [259…
```
combined_df.replace({'Effectiveness Of Internet Usage': {'Very Good':'Very Ef
                                                         'Average':'Somewhat E
```

In [260…
```
combined_df['Effectiveness Of Internet Usage'].value_counts()
```

Out[260…
```
Very Effective        280
Effective             196
Somewhat Effective    186
Not at all             37
Name: Effectiveness Of Internet Usage, dtype: int64
```

Check Devices Used For Internet Browsing

In [261…
```
combined_df['Medium'].value_counts()
```

Out[261…
```
Mobile               420
Laptop and Mobile    215
Desktop               51
Laptop                13
Name: Medium, dtype: int64
```

In [262…
```
combined_df.rename(columns={
    'Medium':'Devices Used For Internet Browsing',
}, inplace=True)

combined_df.columns
```

Out[262…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location', 'Household Internet Facilities', 'Browse Time',
       'Browsing Status', 'Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose of Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority of Learning', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
```

dtype='object')

Check Location Of Internet Use

In [263…
```
combined_df['Location'].value_counts()
```

Out[263…
```
Home          427
University    119
Library        67
Cyber Cafe     59
College        12
School          9
Others          6
Name: Location, dtype: int64
```

In [264…
```
combined_df.rename(columns={
    'Location':'Location Of Internet Use'
}, inplace=True)

combined_df.columns
```

Out[264…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Browse Time', 'Browsing Status', 'Residence',
       'Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
       'Purpose of Use', 'Years of Internet Use', 'Browsing Purpose',
       'Webinar', 'Priority of Learning',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Household Internet Facilities

In [265…
```
combined_df['Household Internet Facilities'].value_counts()
```

Out[265…
```
Not Connected    395
Connected        304
Name: Household Internet Facilities, dtype: int64
```

Check Time Of Internet Browsing

In [266…
```
combined_df['Browse Time'].value_counts()
```

Out[266…
```
Night       446
Day         108
Midnight     74
Morning      71
Name: Browse Time, dtype: int64
```

In [267…
```
combined_df.rename(columns={
       'Browse Time':'Time Of Internet Browsing',
}, inplace=True)

combined_df.columns
```

Out[267…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
```

```
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Browsing Status', 'Residence',
       'Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
       'Purpose of Use', 'Years of Internet Use', 'Browsing Purpose',
       'Webinar', 'Priority of Learning',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Frequency Of Internet Usage

In [268…
```
combined_df['Browsing Status'].value_counts()
```

Out[268…
```
Daily      518
Weekly     162
Monthly     19
Name: Browsing Status, dtype: int64
```

In [269…
```
combined_df.rename(columns={
    'Browsing Status':'Frequency Of Internet Usage',
}, inplace=True)

combined_df.columns
```

Out[269…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage', 'Residence
',
       'Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
       'Purpose of Use', 'Years of Internet Use', 'Browsing Purpose',
       'Webinar', 'Priority of Learning',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Place Of Student's Residence

In [270…
```
combined_df['Residence'].value_counts()
```

Out[270…
```
Town       538
Village    141
Remote      20
Name: Residence, dtype: int64
```

In [271…
```
combined_df.rename(columns={
    'Residence':'Place Of Student\'s Residence',
}, inplace=True)

combined_df.columns
```

Out[271…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage',
       'Place Of Student's Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose of Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority of Learning', 'Internet Usage For Educational Purpose',
```

```
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

Check Purpose Of Internet Use

In [272…
```
combined_df['Purpose of Use'].value_counts()
```

Out[272…
```
Education          286
Social Media       130
Entertainment      118
News                68
Online Shopping     64
Blog                33
Name: Purpose of Use, dtype: int64
```

In [273…
```
combined_df.rename(columns={
    'Purpose of Use':'Purpose Of Internet Use',
}, inplace=True)

combined_df.columns
```

Out[273…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage',
       'Place Of Student's Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose Of Internet Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority of Learning', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

Check Browsing Purpose

In [274…
```
combined_df['Browsing Purpose'].value_counts()
```

Out[274…
```
Academic        433
Non-academic    266
Name: Browsing Purpose, dtype: int64
```

Check Webinar

In [275…
```
combined_df['Webinar'].value_counts()
```

Out[275…
```
No     462
Yes    237
Name: Webinar, dtype: int64
```

Check Priority Of Learning On The Internet

In [276…
```
combined_df['Priority of Learning'].value_counts()
```

Out[276…
```
Academic Learning                  189
Communication Skills               124
Non-academic Learning              122
Leadership Development              88
Creativity and Innovative Skills    88
Career Opportunity                  88
```

```
Name: Priority of Learning, dtype: int64
```

In [277…
```
combined_df.rename(columns={
    'Priority of Learning':'Priority Of Learning On The Internet',
}, inplace=True)

combined_df.columns
```

Out[277…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage',
       'Place Of Student's Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose Of Internet Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority Of Learning On The Internet',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Internet Usage For Educational Purpose

In [278…
```
combined_df['Internet Usage For Educational Purpose'].value_counts()
```

Out[278…
```
Articles or Blogs related to academical studies        157
Notes or lectures for academical purpose               146
Articles or Blogs related to non-academical studies    131
E-books or other Media files                           117
Courses Available on specific topics                    99
Research/Journal/Conference Papers                      49
Name: Internet Usage For Educational Purpose, dtype: int64
```

Check Academic Performance

In [279…
```
combined_df['Academic Performance'].value_counts()
```

Out[279…
```
Good                296
Average             166
Satisfactory        161
Not Satisfactory     76
Name: Academic Performance, dtype: int64
```

In [280…
```
combined_df.replace({'Academic Performance': {'Good':'Excellent', 'Satisfacto
```

In [281…
```
combined_df['Academic Performance'].value_counts()
```

Out[281…
```
Excellent           296
Average             166
Good                161
Not Satisfactory     76
Name: Academic Performance, dtype: int64
```

Check Barriers To Internet Access

In [282…
```
combined_df['Obstacles'].value_counts()
```

Out[282…
```
High Price        333
Bad Service       291
```

```
Unavailability      75
```

In [283…
```
combined_df.rename(columns={
    'Obstacles':'Barriers To Internet Access',
}, inplace=True)

combined_df.columns
```

Out[283…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage',
       'Place Of Student's Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose Of Internet Use',
       'Years of Internet Use', 'Browsing Purpose', 'Webinar',
       'Priority Of Learning On The Internet',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Barriers To Internet Access'],
      dtype='object')
```

Plot the data

Now we can plot the data. Let's write a couple of functions so that we easily plot the data.

This function saves the figures.

In [284…
```
# Write a function to save the figures
PROJECT_ROOT_DIR = "."
DATASET_ID = "Combined"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "Figures", DATASET_ID)
os.makedirs(IMAGES_PATH, exist_ok = True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

This function plots histogram and box plot of the given non-categorical data.

In [285…

```
def numerical_data_plot(dataframe, fig_id, hist_alpha=0.6, color='crimson',
                        title='Image Title', xlabel='X Label', ylabel='Y Label

#     plt.figure(figsize=(10, 6))
#     sns.set(font_scale=1.5)

#     plt.subplot(121)

    count, bin_edges = np.histogram(dataframe)
    dataframe.plot(kind='hist', alpha=hist_alpha,
                   xticks=bin_edges, color=color)

    # Let's add a KDE plot
#     mn, mx = plt.xlim()
#     plt.xlim(mn, mx)
#     kde_x = np.linspace(mn, mx, 300)
#     kde = st.gaussian_kde(dataframe)
#     plt.plot(kde_x, kde.pdf(kde_x) * kde_mul, 'k--', color=color)
#     kde_mul=1000,

#     plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)

#     plt.subplot(122)
#     red_circle = dict(markerfacecolor='r', marker='o')
#     dataframe.plot(kind='box', color=color, flierprops=red_circle)

#     save_fig(fig_id)
```

This function plots histograms of the given categorical data.

In [286…

```
def categorical_bar_plot(dataframe, rot=0, alpha=0.80, color = ['steelblue',
                         title='Distribution', xlabel = 'Column name', ylabel=

    dataframe.value_counts().plot(kind='bar', rot=rot, alpha=alpha, color=colo

    plt.title(title, fontweight='bold')
    plt.xlabel(xlabel, fontweight='bold')
    plt.ylabel(ylabel, fontweight='bold')
```

let's define a function to create scatter plots of the numerical values and check the distribution of the attribute values against the target column, `Academic Performance`

In [287…

```
def categorical_scatter_plot(dataframe, x_column, y_column, title, legend_titl
                             y_label, x_label = 'Number of students'):

    plt.figure(figsize=(15, 7))
    sns.set(font_scale=1.5)
    sns.set_style("whitegrid", {'axes.grid' : False})

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'Excellent'].index
             dataframe[x_column].loc[dataframe[y_column] == 'Excellent'],
             'bo', label = 'Excellent')

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'Good'].index,
             dataframe[x_column].loc[dataframe[y_column] == 'Good'],
             'yo', label = 'Good')

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'Average'].index,
             dataframe[x_column].loc[dataframe[y_column] == 'Average'],
             'go', label = 'Average')

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'Not Satisfactory
             dataframe[x_column].loc[dataframe[y_column] == 'Not Satisfactory
             'ro', label = 'Not Satisfactory')

#     plt.title(title, fontweight='bold')
    plt.xlabel(x_label, fontweight='bold')
    plt.ylabel(y_label, fontweight='bold')
    plt.legend(title = legend_title, title_fontsize=14, loc='lower right', fo
```

let's define a function to create scatter plots of the numerical values and check the distribution of the attribute values against the target column, `Academic Institution`

In [288…

```
def categorical_scatter_plot_against_institution(dataframe, x_column, y_columm
                             y_label, x_label = 'Number of students'):

    plt.figure(figsize=(15, 7))
    sns.set(font_scale=1.5)
    sns.set_style("whitegrid", {'axes.grid' : False})

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'University'].inde
             dataframe[x_column].loc[dataframe[y_column] == 'University'],
             'bo', label = 'University')

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'College'].index,
             dataframe[x_column].loc[dataframe[y_column] == 'College'],
             'go', label = 'College')

    plt.plot(dataframe[x_column].loc[dataframe[y_column] == 'School'].index,
             dataframe[x_column].loc[dataframe[y_column] == 'School'],
             'ro', label = 'School')

#     plt.title(title, fontweight='bold')
    plt.xlabel(x_label, fontweight='bold')
    plt.ylabel(y_label, fontweight='bold')
    plt.legend(title = legend_title, loc='upper right', fontsize=14)
```

A modification of the previous function to create scatter plots of the numerical values vs numerical values and check the distribution of the attribute values against the target

column, `Academic Performance`

In [289…
```
def categorical_scatter_plot_wrt_academic_performance(dataframe, x_column, y_c
                            y_label, x_label, legend_title):

    plt.figure(figsize=(15, 7))
    sns.set(font_scale=1.2)
    sns.set_style("whitegrid", {'axes.grid' : False})

    plt.plot(dataframe[x_column].loc[dataframe['Academic Performance'] == 'Exc
             dataframe[y_column].loc[dataframe['Academic Performance'] == 'Exc
             'bo', label = 'Excellent')

    plt.plot(dataframe[x_column].loc[dataframe['Academic Performance'] == 'Goo
             dataframe[y_column].loc[dataframe['Academic Performance'] == 'Goo
             'yo', label = 'Good')

    plt.plot(dataframe[x_column].loc[dataframe['Academic Performance'] == 'Ave
             dataframe[y_column].loc[dataframe['Academic Performance'] == 'Ave
             'go', label = 'Average')

    plt.plot(dataframe[x_column].loc[dataframe['Academic Performance'] == 'Not
             dataframe[y_column].loc[dataframe['Academic Performance'] == 'Not
             'ro', label = 'Not Satisfactory')

#     plt.title(title, fontweight='bold')
    plt.xlabel(x_label, fontweight='bold')
    plt.ylabel(y_label, fontweight='bold')
    plt.legend(title = legend_title, loc='upper right', fontsize=14)
```

A modification of the previous function to create scatter plots of the numerical values vs numerical values and check the distribution of the attribute values against the target column, `Academic Institution` .

In [290…
```
def categorical_scatter_plot_wrt_academic_institution(dataframe, x_column, y_c
                            y_label, x_label, legend_title):

    plt.figure(figsize=(15, 7))
    sns.set(font_scale=1.2)
    sns.set_style("whitegrid", {'axes.grid' : False})

    plt.plot(dataframe[x_column].loc[dataframe['Academic Institution'] == 'Uni
             dataframe[y_column].loc[dataframe['Academic Institution'] == 'Uni
             'bo', label = 'University')

    plt.plot(dataframe[x_column].loc[dataframe['Academic Institution'] == 'Sch
             dataframe[y_column].loc[dataframe['Academic Institution'] == 'Sch
             'go', label = 'School')

    plt.plot(dataframe[x_column].loc[dataframe['Academic Institution'] == 'Col
             dataframe[y_column].loc[dataframe['Academic Institution'] == 'Col
             'ro', label = 'College')

#     plt.title(title, fontweight='bold')
    plt.xlabel(x_label, fontweight='bold')
    plt.ylabel(y_label, fontweight='bold')
    plt.legend(title = legend_title, loc='upper right', fontsize=14)
```

This function plot histograms of the categorical values against the `'Academic Performance'` column.

These are helper functions.

In [291…
```
def init_dictionary(dictionary, labels):
    for label in labels:
        dictionary[label] = []

def append_to_dict(dictionary, indexes, values):
    x = 0
    for index in indexes:
        dictionary[index].append(values[x])
        x += 1

def furnish_the_lists(labels, indexes, values):
    list_dif = [i for i in labels + indexes if i not in labels or i not in inc

    indexes.extend(list_dif)
    for i in range(len(list_dif)):
        values.append(0)

def append_dataframe_to_dict(dataframe, column_name, labels, dictionary):
    values = dataframe[column_name].value_counts().tolist()
    indexes = dataframe[column_name].value_counts().index.tolist()
    furnish_the_lists(labels, indexes, values)
    append_to_dict(dictionary, indexes, values)

    return dictionary
```

This is the main function.

In [292…
```
def cat_vs_cat_bar_plot(dataframe, column_name, column_cat_list):
    excellent_result_df = dataframe.loc[dataframe['Academic Performance'] ==
    good_result_df = dataframe.loc[dataframe['Academic Performance'] == 'Good
    average_result_df = dataframe.loc[dataframe['Academic Performance'] == 'Av
    unsatisfactory_result_df = dataframe.loc[dataframe['Academic Performance']

    labels = column_cat_list
    dictionary = {}

    init_dictionary(dictionary, labels)

    dictionary = append_dataframe_to_dict(excellent_result_df, column_name, la
    dictionary = append_dataframe_to_dict(good_result_df, column_name, labels,
    dictionary = append_dataframe_to_dict(average_result_df, column_name, labe
    dictionary = append_dataframe_to_dict(unsatisfactory_result_df, column_nar

    return dictionary
```

The following function does the same thing with respect to `'Academic Institution'`.

In [293…
```
def cat_vs_cat_bar_plot_academic_institution(dataframe, column_name, column_ca
    good_result_df = dataframe.loc[dataframe['Academic Institution'] == 'Unive
    satisfactory_result_df = dataframe.loc[dataframe['Academic Institution'] =
    average_result_df = dataframe.loc[dataframe['Academic Institution'] == 'Sc

    labels = column_cat_list
    dictionary = {}

    init_dictionary(dictionary, labels)

    dictionary = append_dataframe_to_dict(good_result_df, column_name, labels,
    dictionary = append_dataframe_to_dict(satisfactory_result_df, column_name,
    dictionary = append_dataframe_to_dict(average_result_df, column_name, labe

    return dictionary
```

The following function does the same thing with respect to `'Browsing Purpose'`

In [294…
```
def cat_vs_cat_bar_plot_browsing_purpose(dataframe, column_name, column_cat_li
    academic_df = dataframe.loc[dataframe['Browsing Purpose'] == 'Academic']
    non_academic_df = dataframe.loc[dataframe['Browsing Purpose'] == 'Non-acad

    labels = column_cat_list
    dictionary = {}

    init_dictionary(dictionary, labels)

    dictionary = append_dataframe_to_dict(academic_df, column_name, labels, di
    dictionary = append_dataframe_to_dict(non_academic_df, column_name, labels

    return dictionary
```

The following function does the same thing with respect to `'Academic Institution'`

In [295…
```
def cat_vs_cat_bar_plot_academic_institution(dataframe, column_name, column_ca
    university_df = dataframe.loc[dataframe['Academic Institution'] == 'Univer
    college_df = dataframe.loc[dataframe['Academic Institution'] == 'College']
    school_df = dataframe.loc[dataframe['Academic Institution'] == 'School']

    labels = column_cat_list
    dictionary = {}

    init_dictionary(dictionary, labels)

    dictionary = append_dataframe_to_dict(university_df, column_name, labels,
    dictionary = append_dataframe_to_dict(college_df, column_name, labels, dic
    dictionary = append_dataframe_to_dict(school_df, column_name, labels, dict

    return dictionary
```

This function add value counts on top of each bar in the histogram.

In [296…
```
def autolabel(rects):

    total_height = 0

    for rect in rects:
        total_height += rect.get_height()

    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format("{:.2f}".format((height/total_height)*100)) +
                    xy = (rect.get_x() + rect.get_width()/2, height),
                    xytext = (0, 3), # 3 points vertical offset
                    textcoords = "offset points",
                    ha = 'center', va = 'bottom')
```

Now let's start plotting the data.

Plotting Non-Categorical Values

Only `'Total Internet Usage(hrs/day)'`, `'Time Spent in Academic(hrs/day)'`, `'Duration Of Internet Usage(In Years)'` are the non-categorical values in the dataset.

Let's plot the bar plot for each of the non-categorical attributes together.

In [297…
```
plt.figure(figsize=(14, 5))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.subplot(121)
numerical_data_plot(combined_df['Total Internet Usage(hrs/day)'], 'Total_Inte
                    title = 'Total internet usage in a day',
                    xlabel = 'Time (hours)', ylabel = 'Number of students')

plt.subplot(122)
numerical_data_plot(combined_df['Time Spent in Academic(hrs/day)'], 'Time_Spe
                    hist_alpha=0.6, color='darkslateblue',
                    title='Total Time spent in academic studies in a day',
                    xlabel='Time (hours)', ylabel='Number of students')


save_fig('Non_Categorical_Bar_plot_collage_1')

plt.show()
```

```
Saving figure Non_Categorical_Bar_plot_collage_1
```

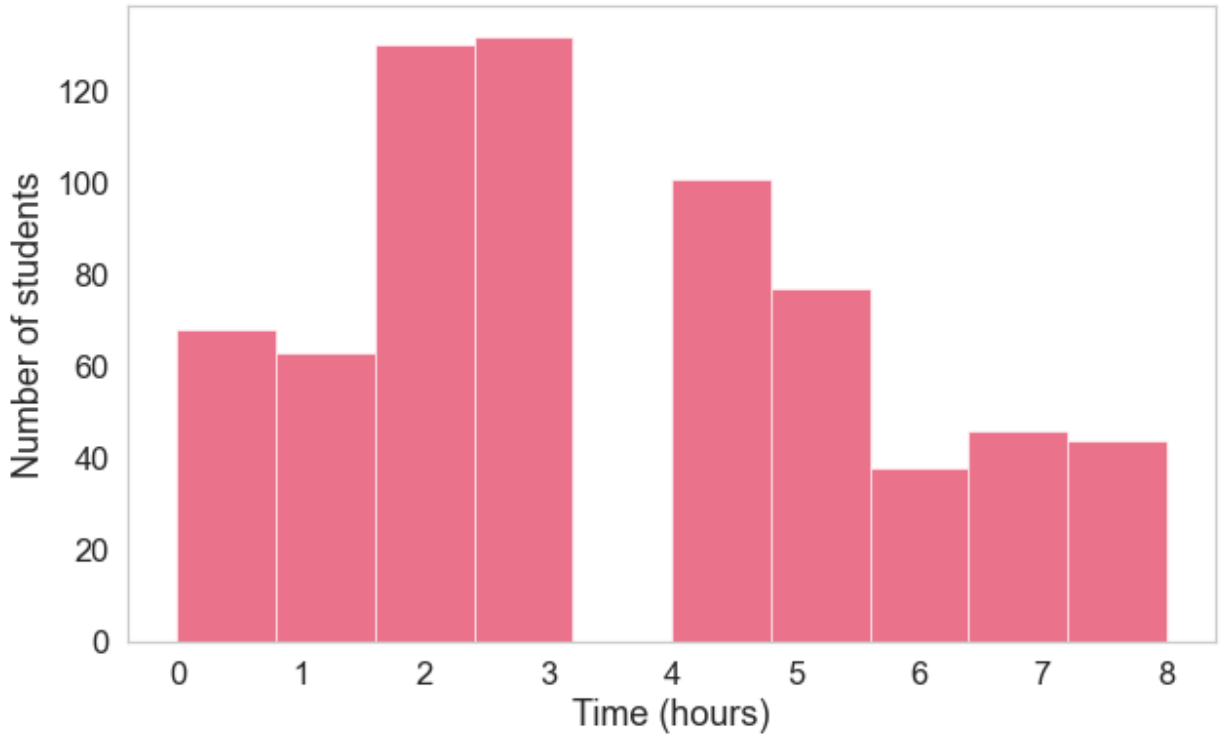
Plotting Total Internet Usage(hrs/day)

In [298…
```
combined_df['Total Internet Usage(hrs/day)'].value_counts()
```

Out[298…
```
3    132
2    130
4    101
5     77
0     68
1     63
7     46
8     44
6     38
Name: Total Internet Usage(hrs/day), dtype: int64
```

First let's check the histogram and the boxplot of this column.

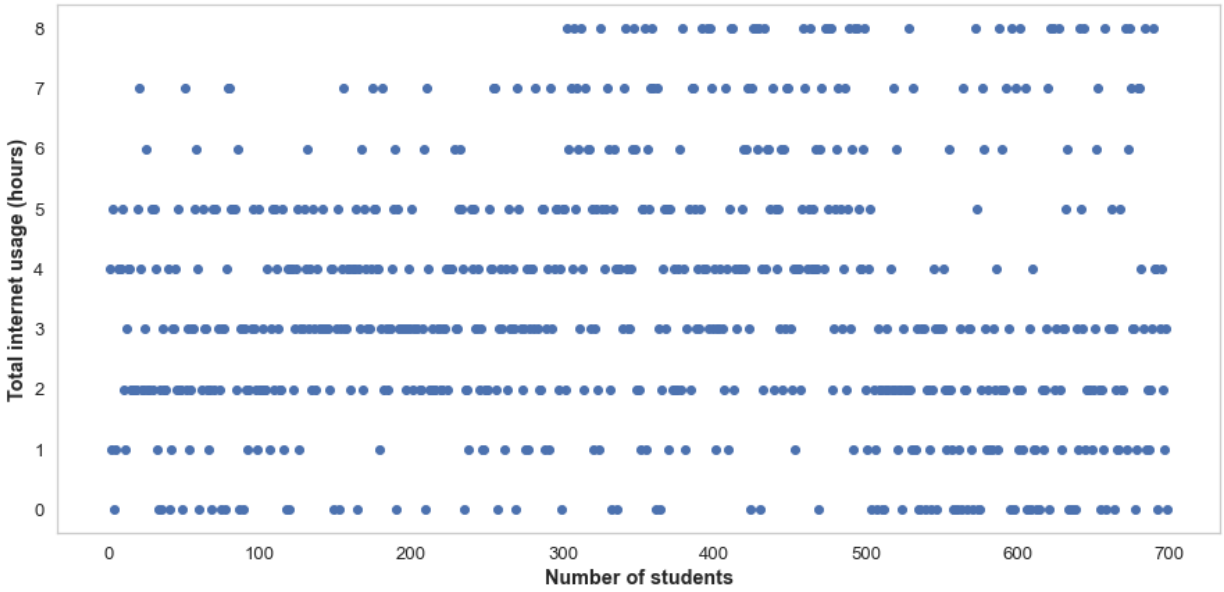
In [299…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

combined_df['Total Internet Usage(hrs/day)'].plot(kind='hist', alpha=0.6, colo

plt.xlabel('Time (hours)')
plt.ylabel('Number of students')

plt.show()
```



Now let's check the scatter plot.

In [300…

```
plt.figure(figsize=(15, 7))
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(np.linspace(1, len(combined_df.index), len(combined_df.index)),
         combined_df['Total Internet Usage(hrs/day)'], 'bo')

plt.ylabel('Total internet usage (hours)', fontweight='bold')
plt.xlabel('Number of students', fontweight='bold')


plt.show()
```



Now let's try plotting `Total Internet Usage(hrs/day)` against the target column `'Academic Performance'` .

In [301…

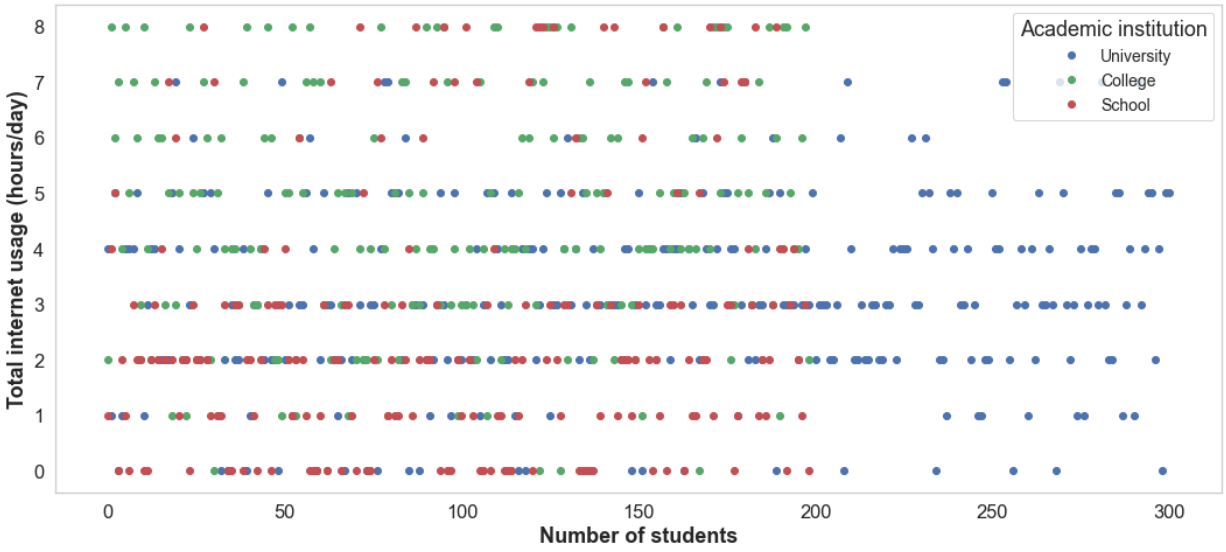
```
categorical_scatter_plot(combined_df, 'Total Internet Usage(hrs/day)', 'Acader
                         'Total Internet Usage In A Day W.R.T. Academic Perforr
                          'Total internet usage (hours/day)')

plt.fill_between([-1, 305], [5.2, 5.2], -0.2, color='steelblue', alpha=0.1, ir
plt.fill_between([-1, 305], [8.1, 8.1], 3, color='green', alpha=0.1, interpola

save_fig('Total_Internet_Usage_vs_Academic_Performance_Scatter_Plot')

plt.show()
```

```
Saving figure Total_Internet_Usage_vs_Academic_Performance_Scatter_Plot
```

Now let's try plotting `Total Internet Usage(hrs/day)` against the target column `'Academic Institution'`.

In [302…
```
categorical_scatter_plot_against_institution(combined_df, 'Total Internet Usag
                          'Total Internet Usage In A Day vs Academic Institutio
                                        'Total internet usage (hours/day)

save_fig('Total_Internet_Usage_In_A_Day_WRT_Academic_Institution_Scatter_Plot
plt.show()
```

```
Saving figure Total_Internet_Usage_In_A_Day_WRT_Academic_Institution_Scatter_P
lot
```



Plotting `Time Spent in Academic(hrs/day)`

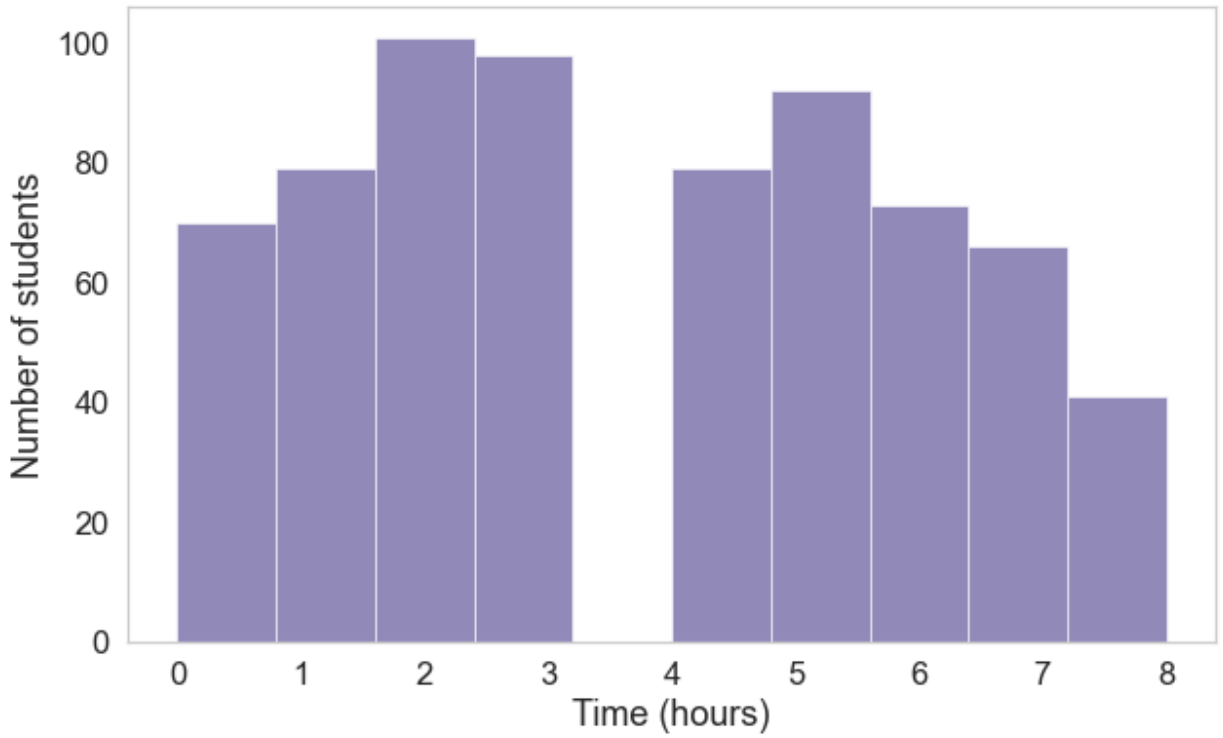
In [303…
```
combined_df['Time Spent in Academic(hrs/day)'].value_counts()
```

Out[303…
```
2    101
3     98
5     92
4     79
1     79
6     73
0     70
7     66
8     41
Name: Time Spent in Academic(hrs/day), dtype: int64
```

First let's check the histogram and the boxplot of this column.

In [304…

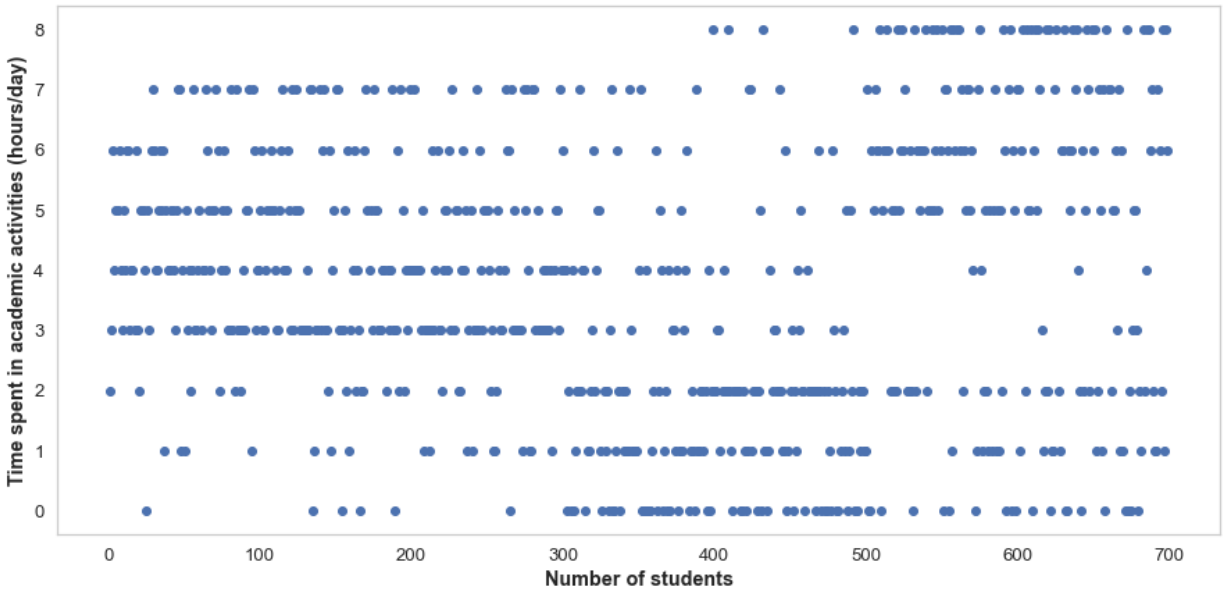
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

combined_df['Time Spent in Academic(hrs/day)'].plot(kind='hist', alpha=0.6, c

# plt.title('Total time spent in academic studies in a day')
plt.xlabel('Time (hours)')
plt.ylabel('Number of students')

plt.show()
```



Now let's check the scatter plot.

In [305…

```
plt.figure(figsize=(15,7))
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(np.linspace(1, len(combined_df.index), len(combined_df.index)),
         combined_df['Time Spent in Academic(hrs/day)'], 'bo')

# plt.title('Total time spent in academic in a day', fontweight='bold')
plt.ylabel('Time spent in academic activities (hours/day)', fontweight='bold')
plt.xlabel('Number of students', fontweight='bold')

plt.show()
```

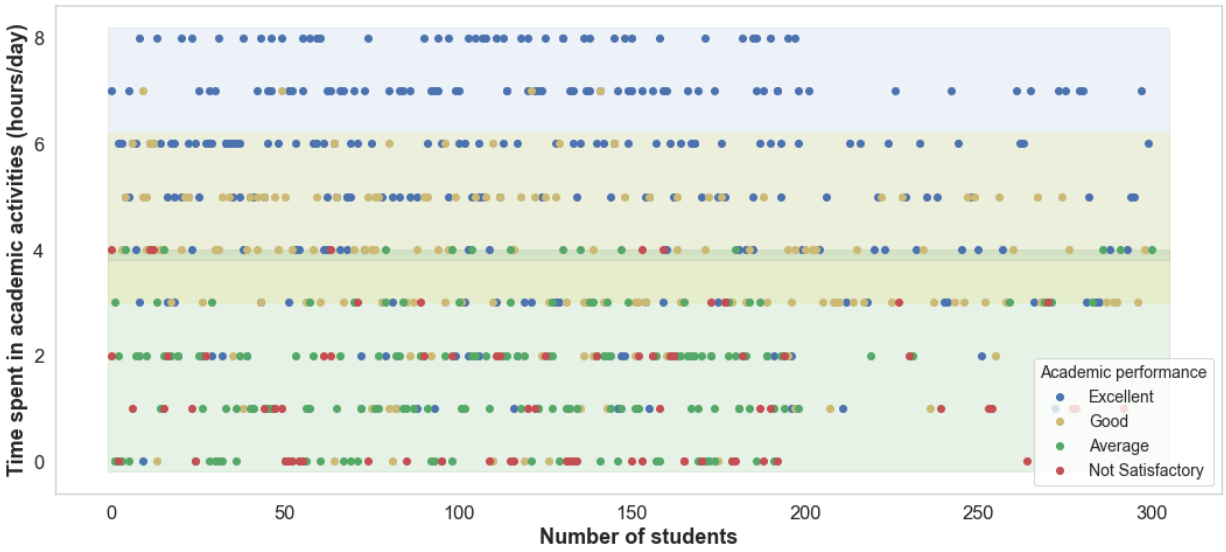


Now let's try plotting `Time Spent in Academic(hrs/day)` against the target column `'Academic Performance'`.

In [306…

```
categorical_scatter_plot(combined_df, 'Time Spent in Academic(hrs/day)', 'Acad
                         'Time Spent In Academic In A Day W.R.T. Academic Perf
                         'Time spent in academic activities (hours/day)')

plt.fill_between([-1, 305], [8.2, 8.2], 3.8, color='steelblue', alpha=0.1, int
plt.fill_between([-1, 305], [6.2, 6.2], 3, color='gold', alpha=0.1, interpolat
plt.fill_between([-1, 305], [4, 4], -0.2, color='green', alpha=0.1, interpolat

save_fig('Time_Spent_in_Academic_vs_Academic_Performance_Scatter_Plot')

plt.show()
```

```
Saving figure Time_Spent_in_Academic_vs_Academic_Performance_Scatter_Plot
```

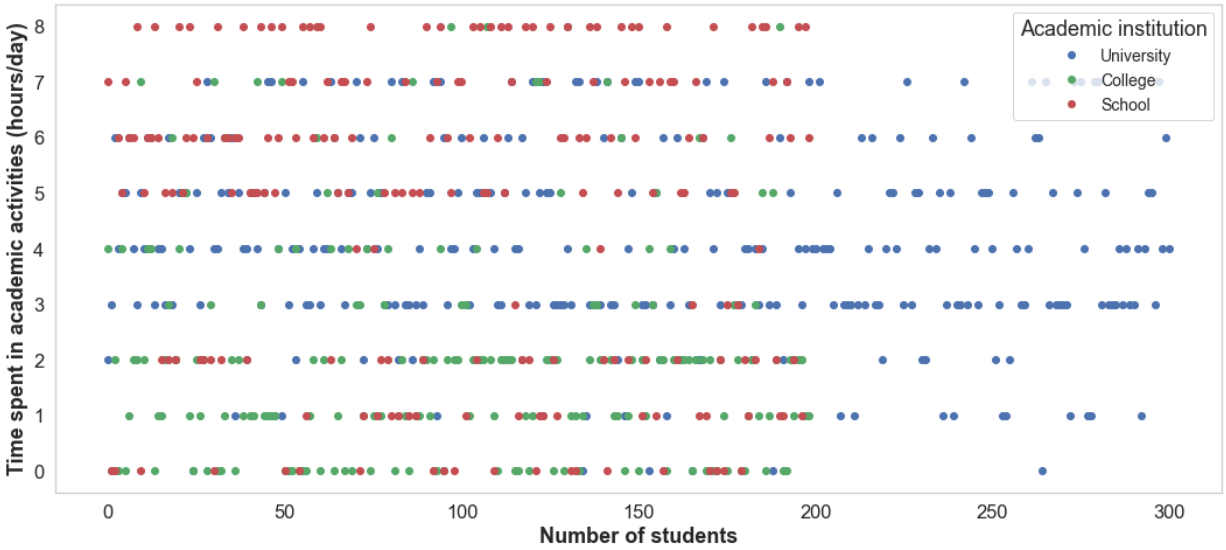


Now let's try plotting `Time Spent in Academic(hrs/day)` against the target column `'Academic Institution'`.

In [307…

```
categorical_scatter_plot_against_institution(combined_df, 'Time Spent in Acade
                            'Time Spent in Academic In A Day vs Academic Institut
                                                'Time spent in academic activitie

save_fig('Time_Spent_in_Academic_vs_Academic_Institution_Scatter_Plot')
plt.show()
```

```
Saving figure Time_Spent_in_Academic_vs_Academic_Institution_Scatter_Plot
```

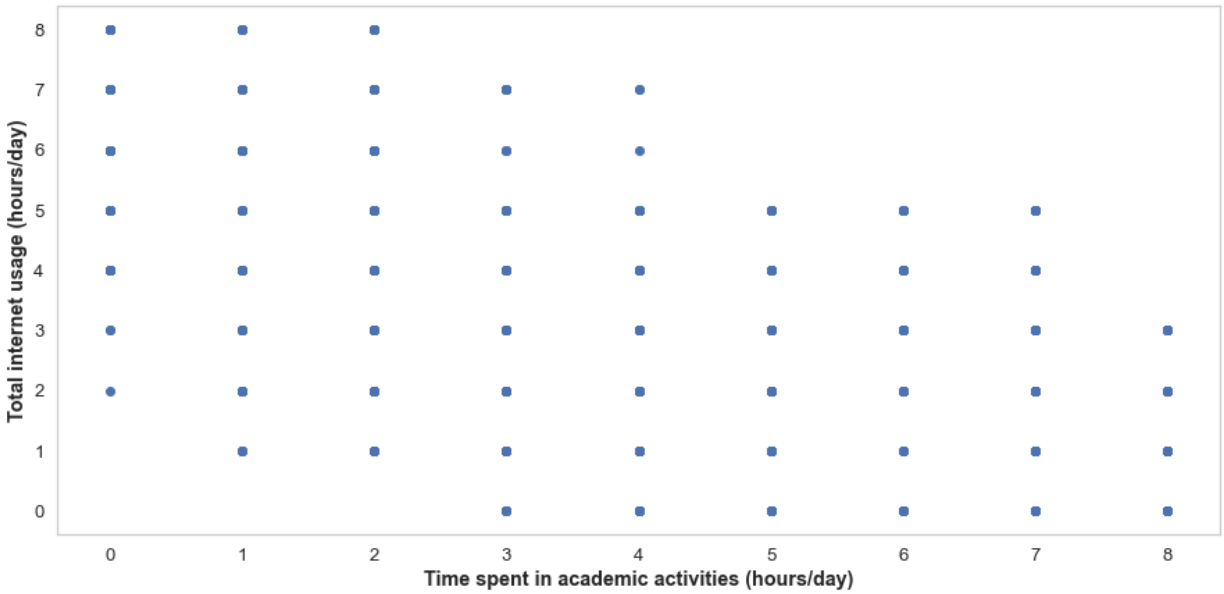


Plotting Time Spent in Academic(hrs/day) vs Total Internet Usage(hrs/day)

Let's use scatter plot.

In [308…

```
plt.figure(figsize=(15, 7))
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(combined_df['Time Spent in Academic(hrs/day)'],
         combined_df['Total Internet Usage(hrs/day)'], 'bo')

# plt.title('Time Spent in Academic(hrs/day) vs Total Internet Usage(hrs/day)
plt.xlabel('Time spent in academic activities (hours/day)', fontweight='bold')
plt.ylabel('Total internet usage (hours/day)', fontweight='bold')

plt.show()
```

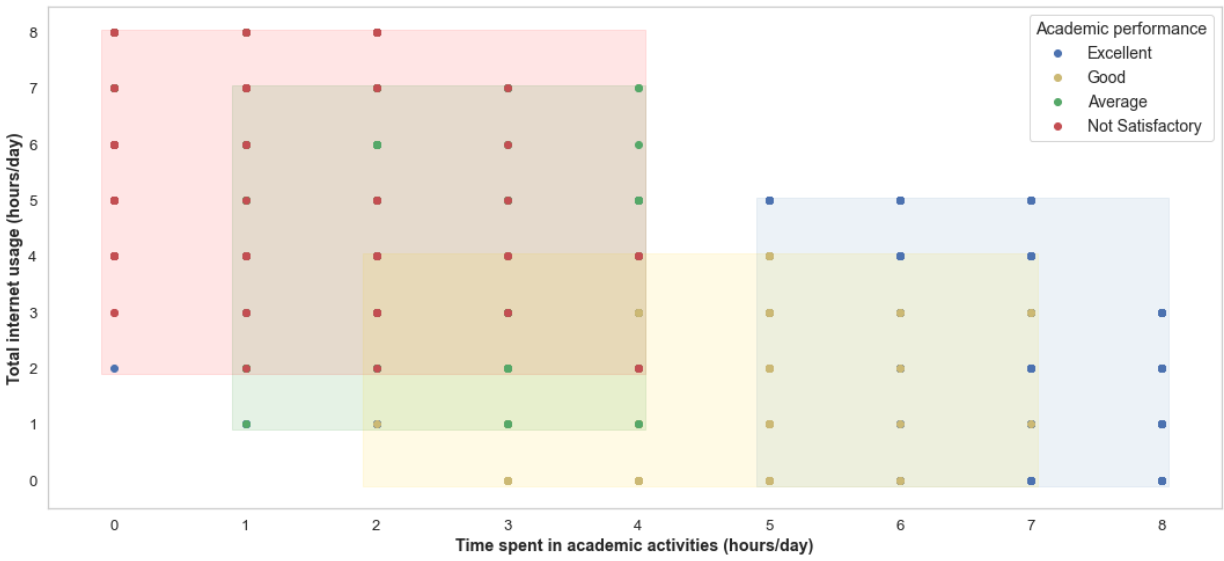


Now let's try plotting `Time Spent in Academic(hrs/day)` vs `'Total Internet Usage(hrs/day)'` against the target `'Academic Performance'`.

In [309…

```
categorical_scatter_plot_wrt_academic_performance(combined_df,  'Time Spent in
                                                  'Total Internet Usage(hrs/da
                                                  'Time Spent in Academic(hrs/
                                                  'Total internet usage (hours
                                                  'Time spent in academic acti
                                                  'Academic performance')


plt.fill_between([-0.1, 4.05], [8.05, 8.05], 1.9, color='red', alpha=0.1, inte
plt.fill_between([0.9, 4.05], [7.05, 7.05], 0.9, color='green', alpha=0.1, int
plt.fill_between([4.9, 8.05], [5.05, 5.05], -0.1, color='steelblue', alpha=0.1
plt.fill_between([1.9, 7.05], [4.05, 4.05], -0.1, color='gold', alpha=0.1, int

save_fig('Time_Spent_in_Academic_vs_Total_Internet_Usage')

plt.show()
```

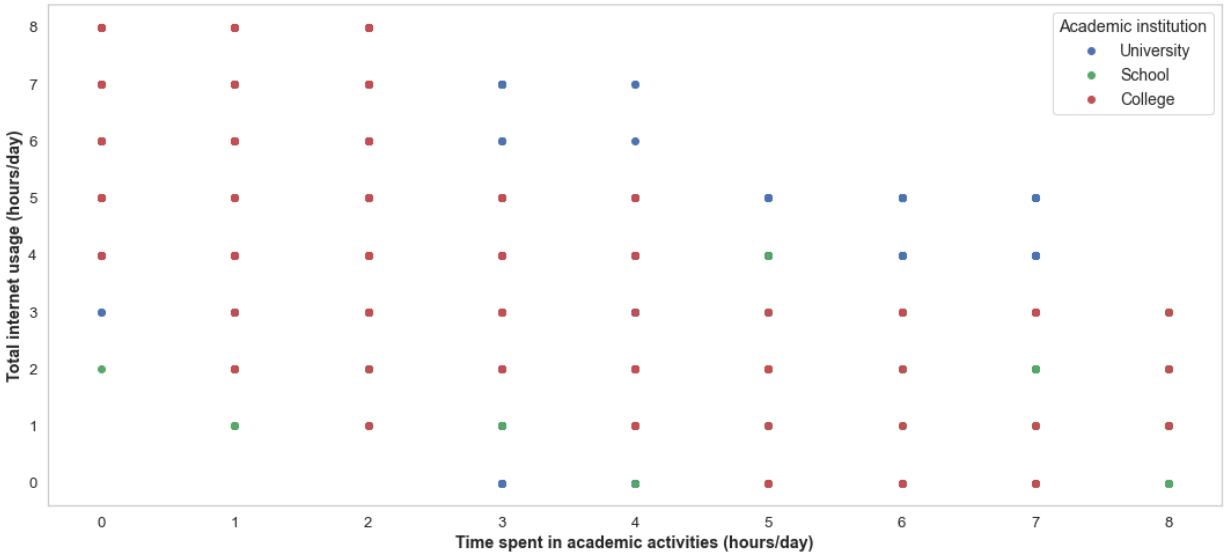
```
Saving figure Time_Spent_in_Academic_vs_Total_Internet_Usage
```



Now let's try plotting `Time Spent in Academic(hrs/day)` vs `'Total Internet Usage(hrs/day)'` against the target `'Academic Institution'`.

In [310…
```
categorical_scatter_plot_wrt_academic_institution(combined_df,  'Time Spent in
                                                  'Total Internet Usage(hrs/da
                                                  'Time Spent in Academic(hrs/
                                                  'Total internet usage (hours
                                                  'Time spent in academic acti
                                                  'Academic institution')


save_fig('Time_Spent_in_Academic_vs_Total_Internet_Usage_Academic_Institution

plt.show()
```

```
Saving figure Time_Spent_in_Academic_vs_Total_Internet_Usage_Academic_Institut
ion
```



Plotting Duration Of Internet Usage(In Years)

In [311…
```
combined_df.rename(columns={
    'Years of Internet Use':'Duration Of Internet Usage(In Years)',
}, inplace=True)

combined_df.columns
```

Out[311…
```
Index(['Gender', 'Age', 'Academic Institution', 'Frequently Visited Website',
       'Effectiveness Of Internet Usage', 'Devices Used For Internet Browsing
',
       'Location Of Internet Use', 'Household Internet Facilities',
       'Time Of Internet Browsing', 'Frequency Of Internet Usage',
       'Place Of Student's Residence', 'Total Internet Usage(hrs/day)',
       'Time Spent in Academic(hrs/day)', 'Purpose Of Internet Use',
       'Duration Of Internet Usage(In Years)', 'Browsing Purpose', 'Webinar',
       'Priority Of Learning On The Internet',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Barriers To Internet Access'],
      dtype='object')
```

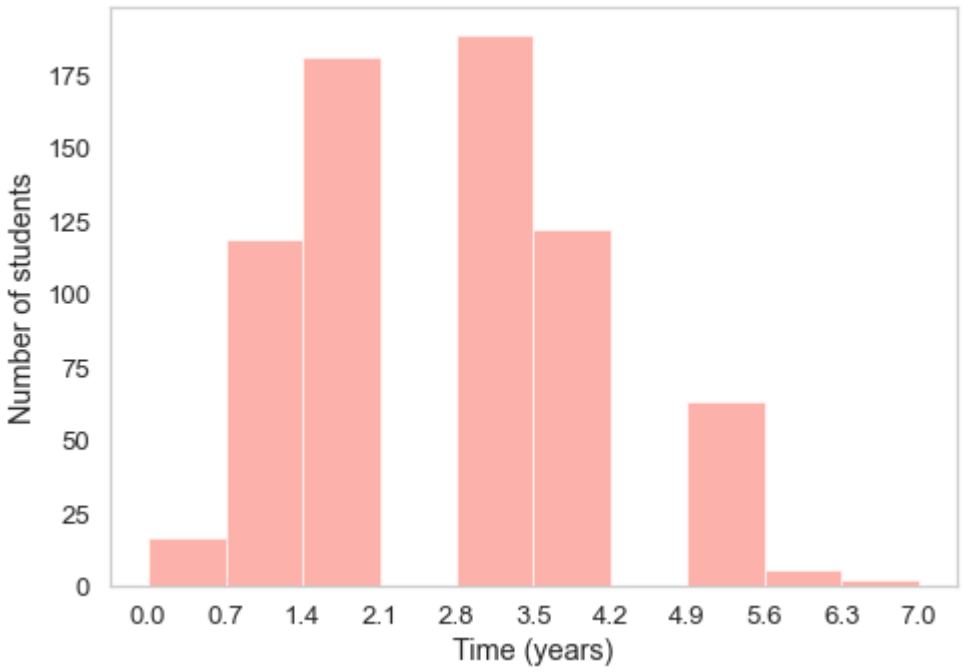
In [312…
```
combined_df['Duration Of Internet Usage(In Years)'].value_counts()
```

Out[312…
```
3    189
2    181
4    122
1    119
5     63
0     17
6      6
7      2
Name: Duration Of Internet Usage(In Years), dtype: int64
```

First let's check the histogram and the boxplot of this column.

In [313…
```
plt.figure(figsize=(7, 5))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

numerical_data_plot(combined_df['Duration Of Internet Usage(In Years)'], 'Dura
                    hist_alpha=0.6, color='salmon',
                    title='How long have the students been using internet?', x
                    ylabel='Number of students')


save_fig('Non_Categorical_Bar_plot_2')

plt.show()
```

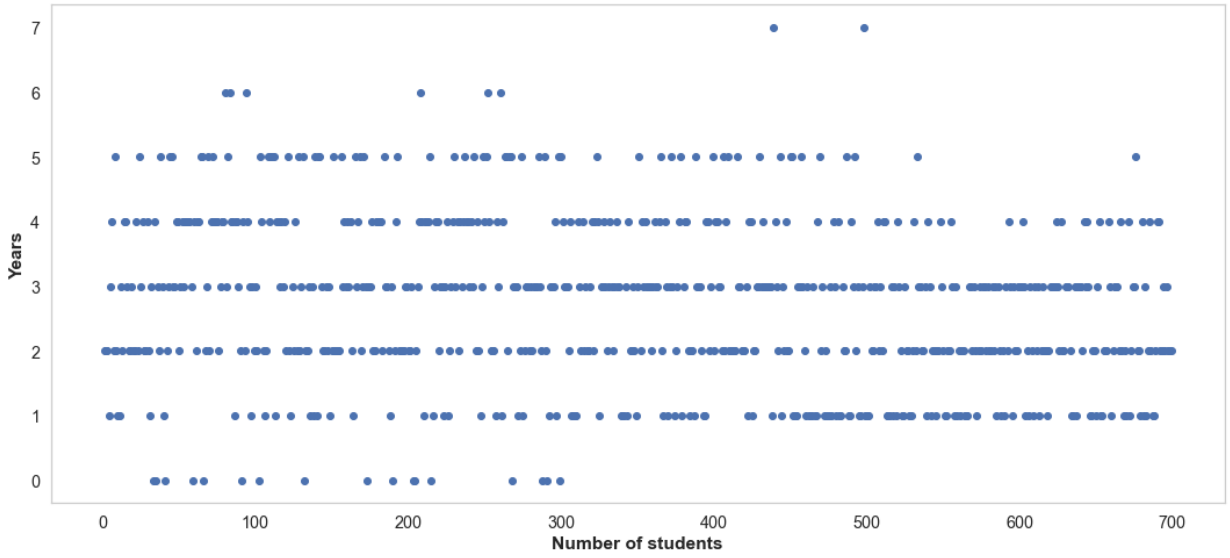
```
Saving figure Non_Categorical_Bar_plot_2
```



Now let's check the scatter plot.

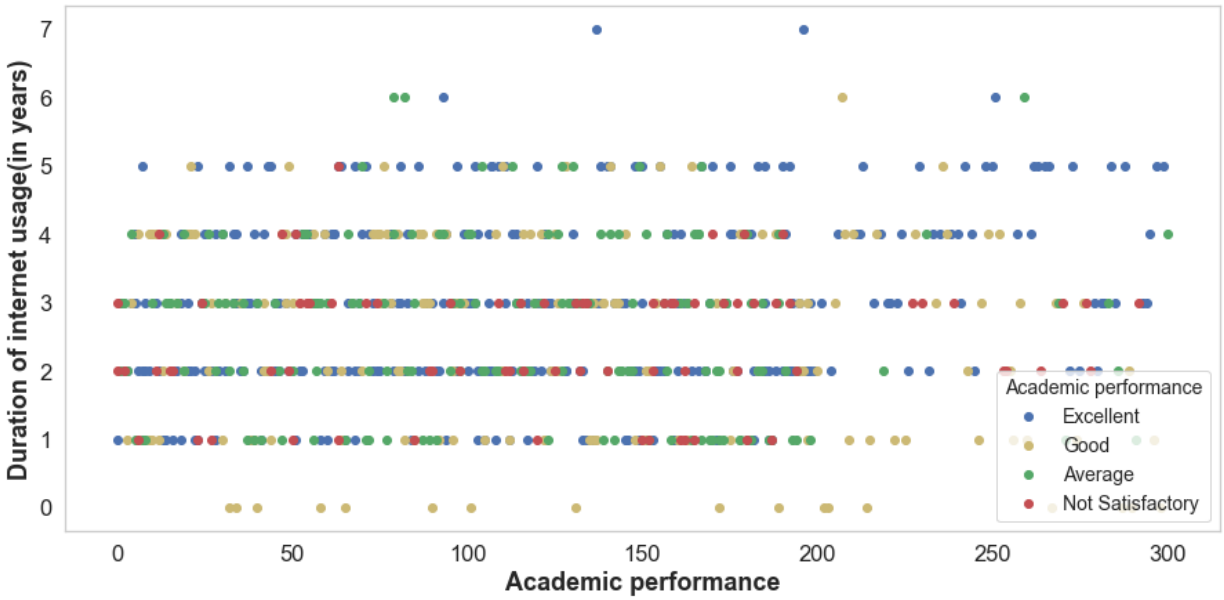
In [314…
```
plt.figure(figsize=(15, 7))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(np.linspace(1, len(combined_df.index), len(combined_df.index)),
         combined_df['Duration Of Internet Usage(In Years)'], 'bo')

# plt.title('Duration Of Internet Usage (In Years)', fontweight='bold')
plt.ylabel('Years', fontweight='bold')
plt.xlabel('Number of students', fontweight='bold')

save_fig('Duration_Of_Internet_Usage_In_Years_Scatter_Plot')
plt.show()
```

```
Saving figure Duration_Of_Internet_Usage_In_Years_Scatter_Plot
```



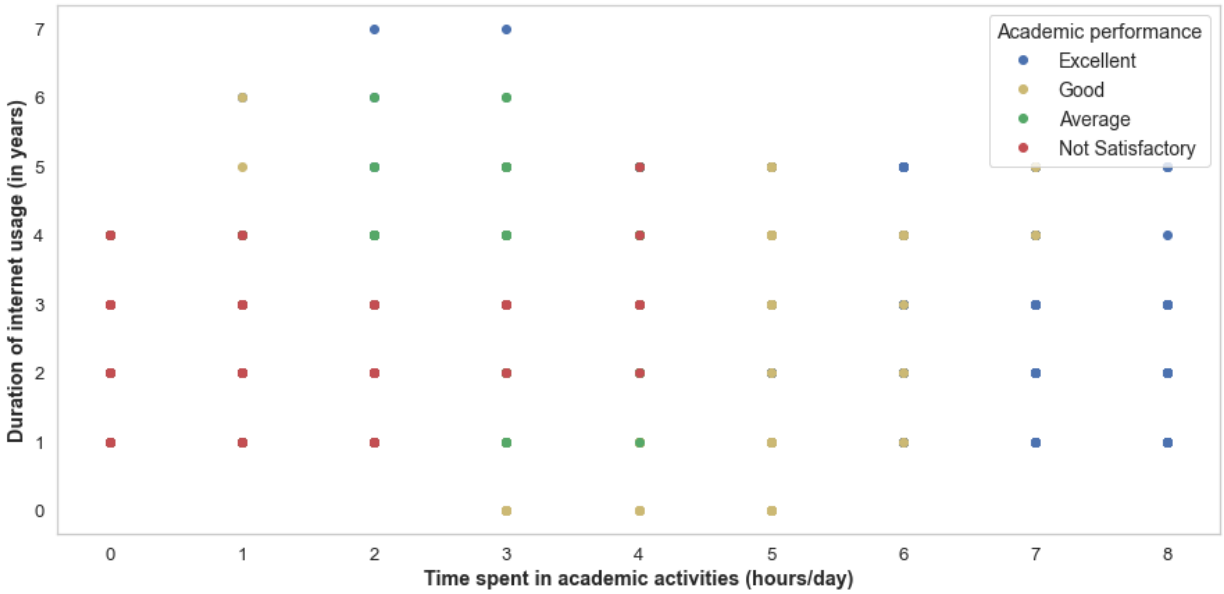
Now let's try plotting `'Years of Internet Use'` against the target column `'Academic Performance'`.

In [315…
```
categorical_scatter_plot(combined_df, 'Duration Of Internet Usage(In Years)',
                         'Duration Of Internet Usage(In Years) vs Academic Pe
                         'Duration of internet usage(in years)', 'Academic pe
```



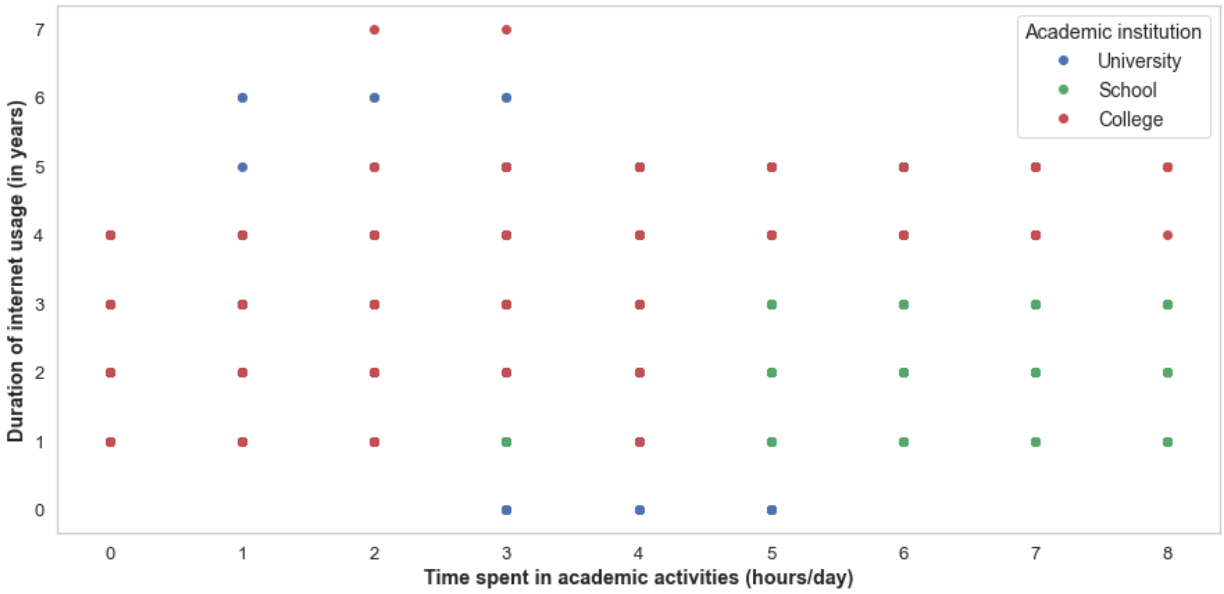
Now let's try plotting `Time Spent in Academic(hrs/day)` vs `'Duration Of Internet Usage(In Years)'` against the target `'Academic Performance'`.

In [316…
```
categorical_scatter_plot_wrt_academic_performance(combined_df, 'Time Spent in
                                                  'Duration Of Internet Usage
                                                  'Time Spent in Academic (hrs
                                                  'Duration of internet usage
                                                  'Time spent in academic acti

plt.show()
```



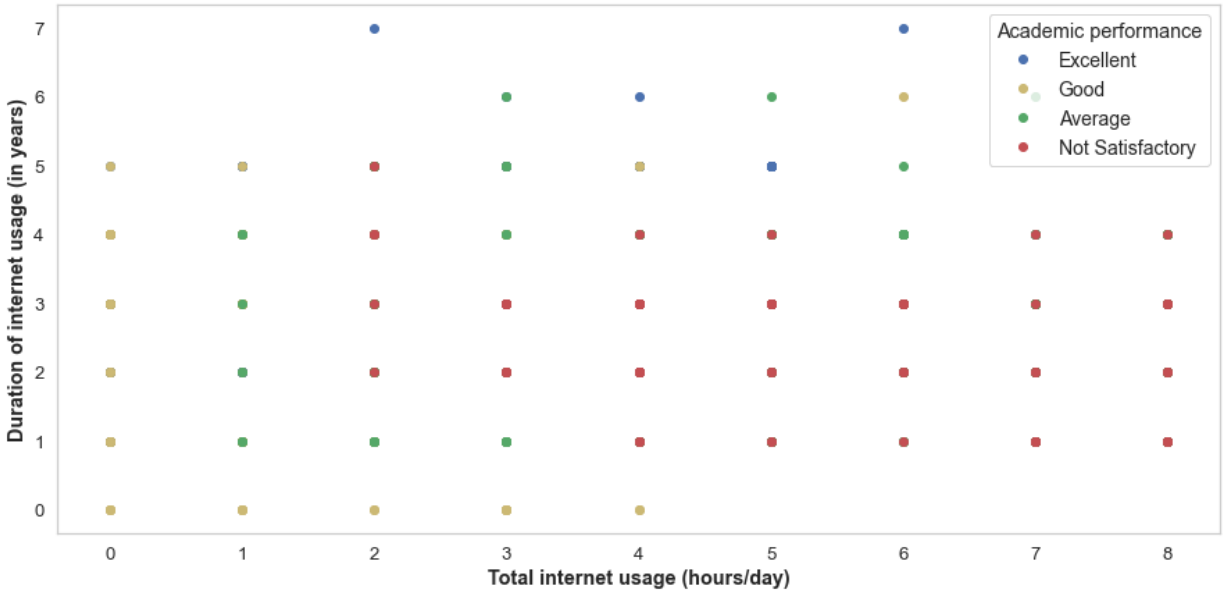
Now let's try plotting `Time Spent in Academic(hrs/day)` vs `'Duration Of Internet Usage(In Years)'` against the target `'Academic Institution'`.

In [317…
```
categorical_scatter_plot_wrt_academic_institution(combined_df, 'Time Spent in
                                                  'Duration Of Internet Usage
                                                  'Time Spent in Academic (hrs
                                                  'Duration of internet usage
                                                  'Time spent in academic acti

plt.show()
```



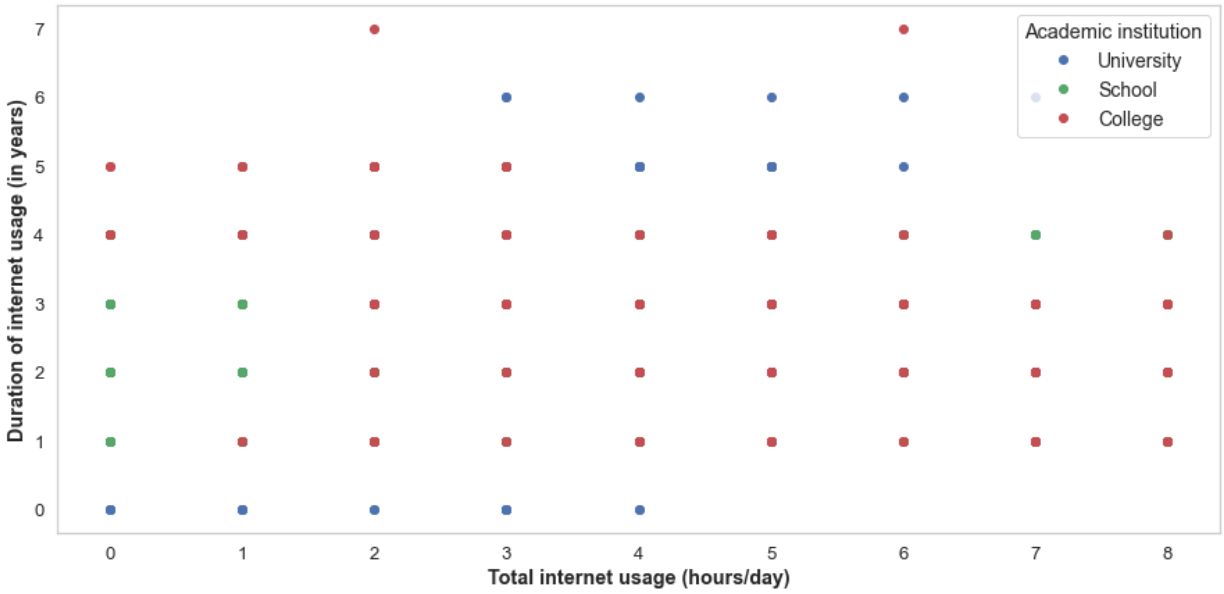
Now let's try plotting `'Total Internet Usage(hrs/day)'` vs `'Duration Of Internet Usage(In Years)'` against the target `'Academic Performance'`.

In [318…
```
categorical_scatter_plot_wrt_academic_performance(combined_df, 'Total Internet
                                                  'Duration Of Internet Usage
                                                  'Total Internet Usage (hrs/
                                                  'Duration of internet usage
                                                  'Total internet usage (hours

plt.show()
```



Now let's try plotting `'Total Internet Usage(hrs/day)'` vs `'Duration Of Internet Usage(In Years)'` against the target `'Academic Institution'`.

In [319…]
```
categorical_scatter_plot_wrt_academic_institution(combined_df, 'Total Internet
                                                  'Duration Of Internet Usage
                                                  'Total Internet Usage (hrs/
                                                  'Duration of internet usage
                                                  'Total internet usage (hours


plt.show()
```



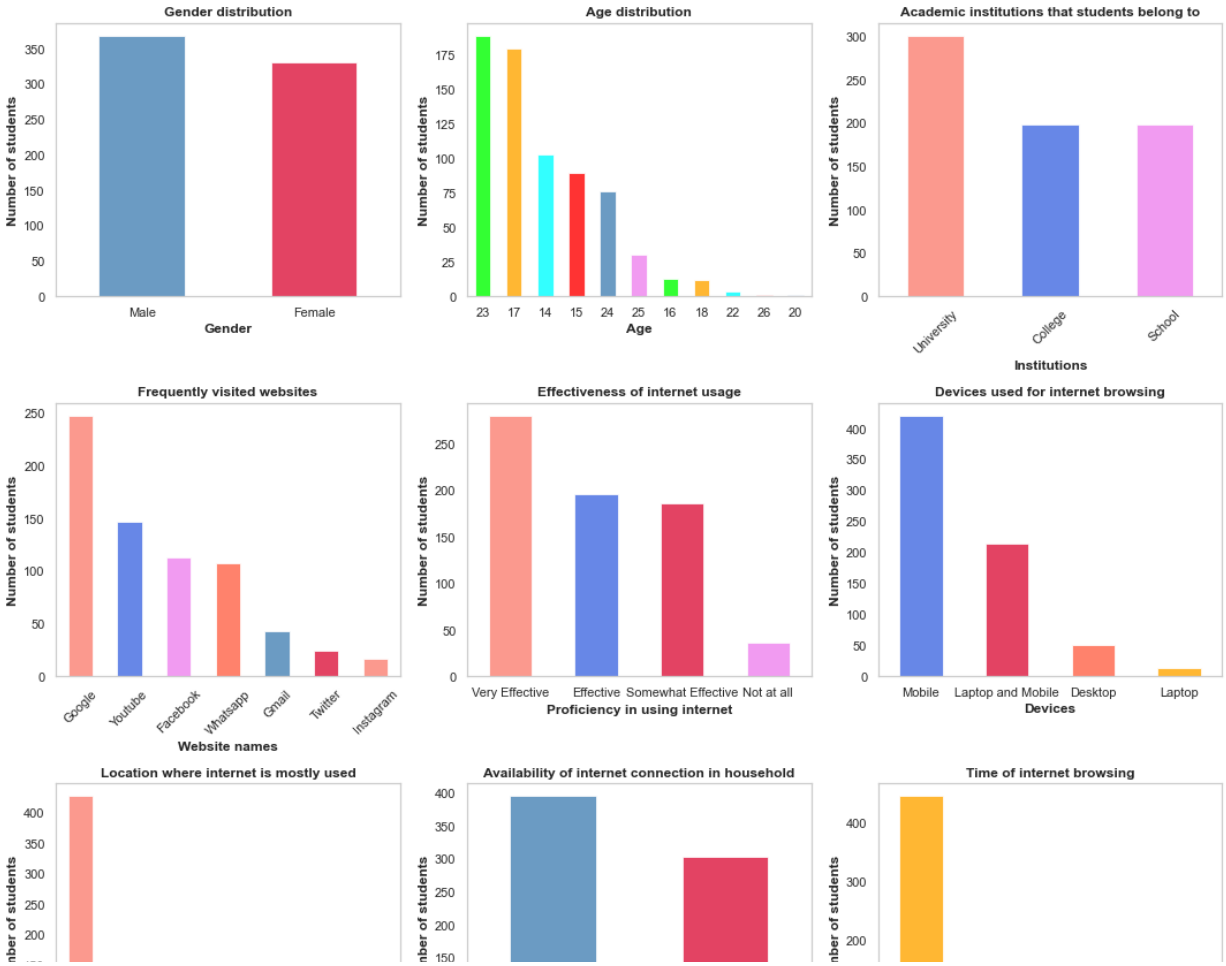
Plotting Categorical Values

`'Gender'`, `'Age'`, `'Academic Institution'`, `'Frequently Visited Website'`, `'Effectiveness Of Internet Usage'`, `'Devices Used For Internet Browsing'`, `'Location Of Internet Use'`, `'Household Internet Facilities'`, `'Time Of Internet Browsing'`, `'Frequency Of Internet Usage'`, `'Place Of Student's Residence'`, `'Purpose Of Internet Use'`, `'Browsing Purpose'`, `'Webinar'`, `'Priority Of Learning On The Internet'`, `'Academic Performance'`, `'Barriers To Internet Access'` are the categorical values in the dataset.

In [320…

```
plt.figure(figsize=(15, 14))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.subplot(331)
categorical_bar_plot(combined_df['Gender'], title='Gender distribution', xlabe

plt.subplot(332)
categorical_bar_plot(combined_df['Age'],
                     color=['lime', 'orange', 'cyan', 'red', 'steelblue', 'vi
                     title='Age distribution', xlabel='Age')

plt.subplot(333)
categorical_bar_plot(combined_df['Academic Institution'], rot=45,
                     color=['salmon', 'royalblue', 'violet', 'tomato', 'steelb
                     title='Academic institutions that students belong to', xl

plt.subplot(334)
categorical_bar_plot(combined_df['Frequently Visited Website'], rot=45,
                     color=['salmon', 'royalblue', 'violet', 'tomato', 'steelb
                     title='Frequently visited websites', xlabel='Website name

plt.subplot(335)
categorical_bar_plot(combined_df['Effectiveness Of Internet Usage'],
                     color=['salmon', 'royalblue', 'crimson', 'violet'],
                     title='Effectiveness of internet usage', xlabel='Proficie

plt.subplot(336)
categorical_bar_plot(combined_df['Devices Used For Internet Browsing'],
                     color=['royalblue', 'crimson', 'tomato', 'orange'],
                     title='Devices used for internet browsing', xlabel='Devic

plt.subplot(337)
categorical_bar_plot(combined_df['Location Of Internet Use'], rot=45,
                     color=['salmon', 'crimson', 'violet', 'orange', 'steelblu
                     title='Location where internet is mostly used', xlabel='L

plt.subplot(338)
categorical_bar_plot(combined_df['Household Internet Facilities'],
                     title='Availability of internet connection in household',
                     xlabel='Household internet facilities')

plt.subplot(339)
categorical_bar_plot(combined_df['Time Of Internet Browsing'], color=['orange
                     title='Time of internet browsing', xlabel='Browsing time


save_fig('Bar_plot_collage_1')

plt.show()
```

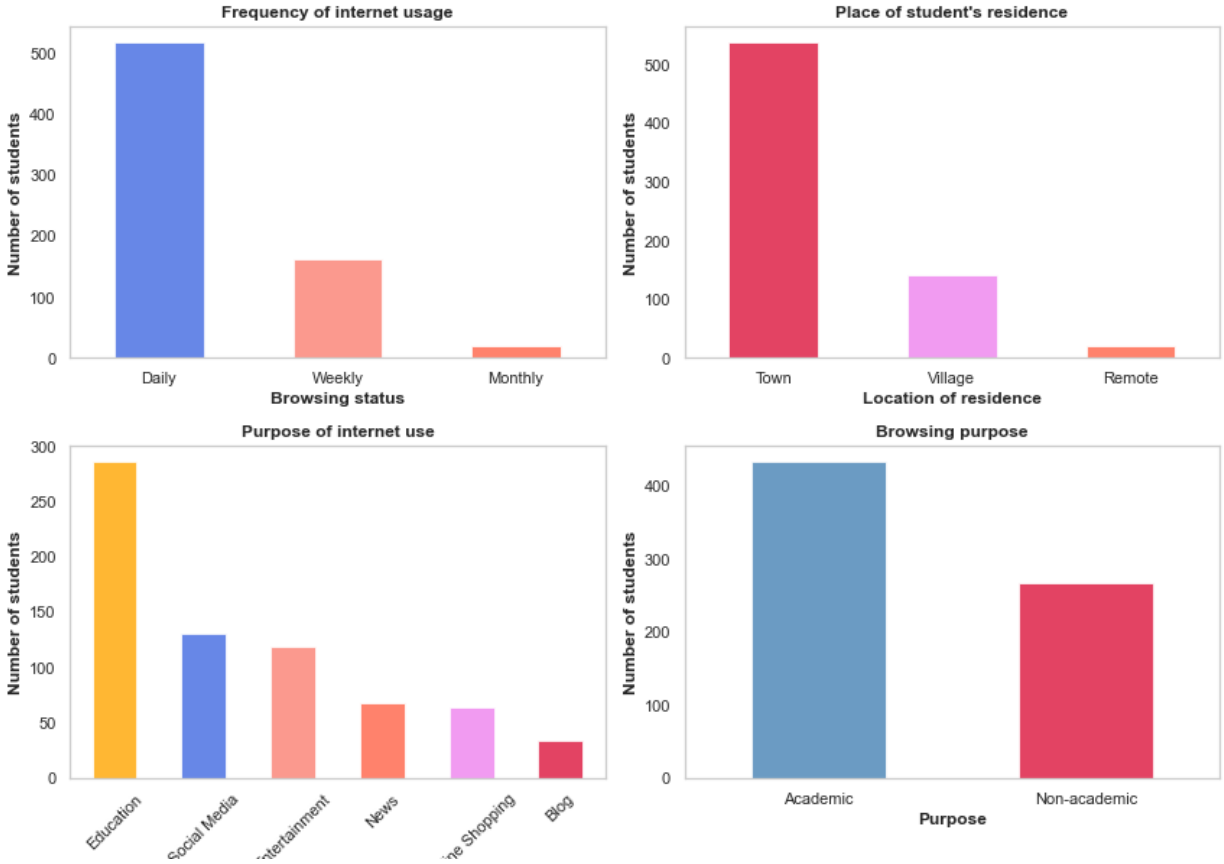
```
Saving figure Bar_plot_collage_1
```



In [321…

```
plt.figure(figsize=(12, 9))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.subplot(221)
categorical_bar_plot(combined_df['Frequency Of Internet Usage'], color=['royal
                     title='Frequency of internet usage', xlabel='Browsing sta
#
plt.subplot(222)
categorical_bar_plot(combined_df['Place Of Student\'s Residence'], color=['cri
                     title='Place of student\'s residence', xlabel='Location
#
plt.subplot(223)
categorical_bar_plot(combined_df['Purpose Of Internet Use'], rot=45,
                     color = ['orange', 'royalblue', 'salmon', 'tomato', 'viol
                     title='Purpose of internet use', xlabel='Purpose of use'
#
plt.subplot(224)
categorical_bar_plot(combined_df['Browsing Purpose'], title='Browsing purpose
                     xlabel='Purpose')


save_fig('Bar_plot_collage_2')
plt.show()
```

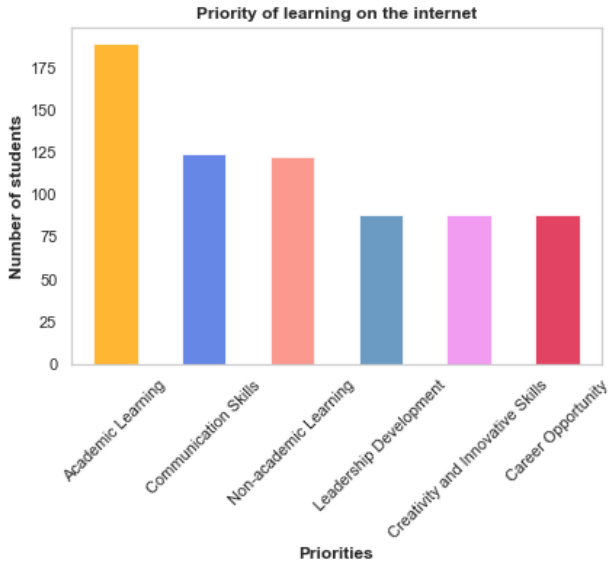
```
Saving figure Bar_plot_collage_2
```

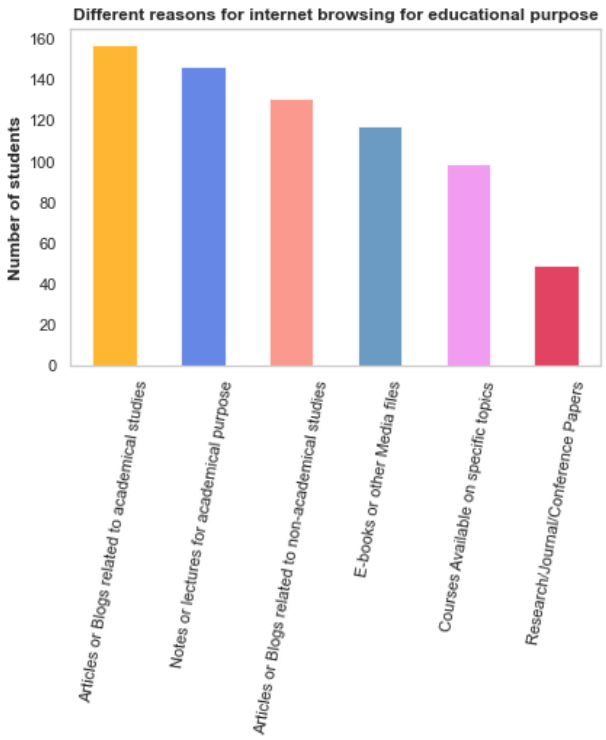


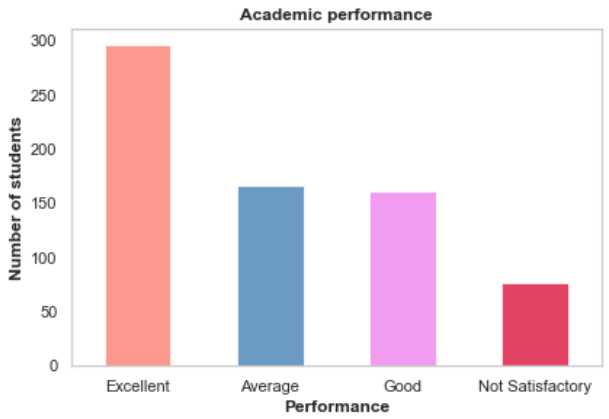
In [322…

```
plt.figure(figsize=(12, 13))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1)
sns.set_style("whitegrid", {'axes.grid' : False})

#
plt.subplot(221)
categorical_bar_plot(combined_df['Webinar'], color=['salmon', 'crimson'],
                     title='Participation in webinars', xlabel='Participation
#
plt.subplot(222)
categorical_bar_plot(combined_df['Priority Of Learning On The Internet'], rot=
                     color = ['orange', 'royalblue', 'salmon', 'steelblue', 'v
                     title='Priority of learning on the internet', xlabel='Pri

#
plt.subplot(223)
categorical_bar_plot(combined_df['Internet Usage For Educational Purpose'], ro
                     color=['orange', 'royalblue', 'salmon', 'steelblue', 'vio
                     title='Different reasons for internet browsing for educat
                     xlabel='Internet Usage For Educational Purpose')

plt.subplot(224)
categorical_bar_plot(combined_df['Academic Performance'], color=['salmon', 'st
                     title='Academic performance', xlabel='Performance')


save_fig('Bar_plot_collage_3')
plt.show()
```

```
Saving figure Bar_plot_collage_3
```



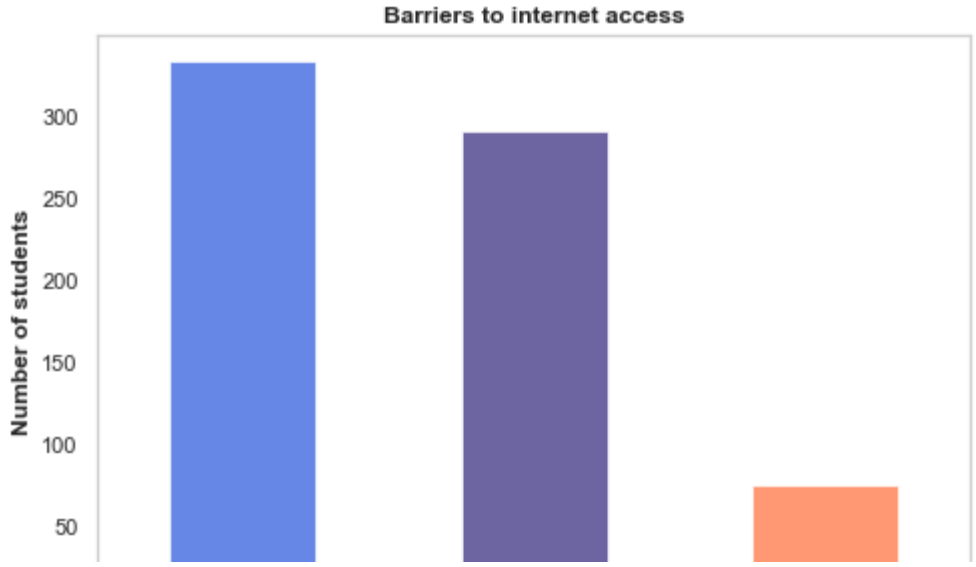




In [323…

```
plt.figure(figsize=(7, 5))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Barriers To Internet Access'],
                     color=['royalblue', 'darkslateblue', 'coral', 'crimson'],
                     title='Barriers to internet access', xlabel='Obstacles')


save_fig('Bar_plot_collage_4')
plt.show()
```

```
Saving figure Bar_plot_collage_4
```

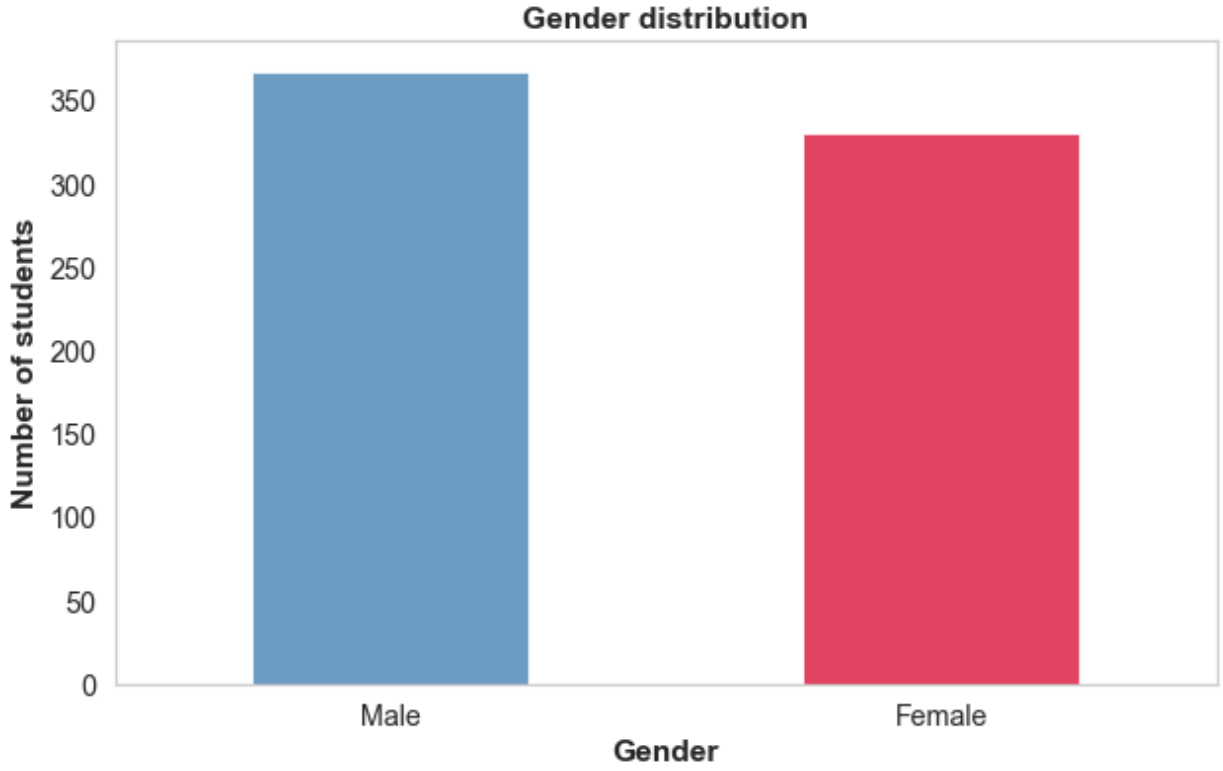


Plotting 'Gender'

Let's check the histogram.

In [324…

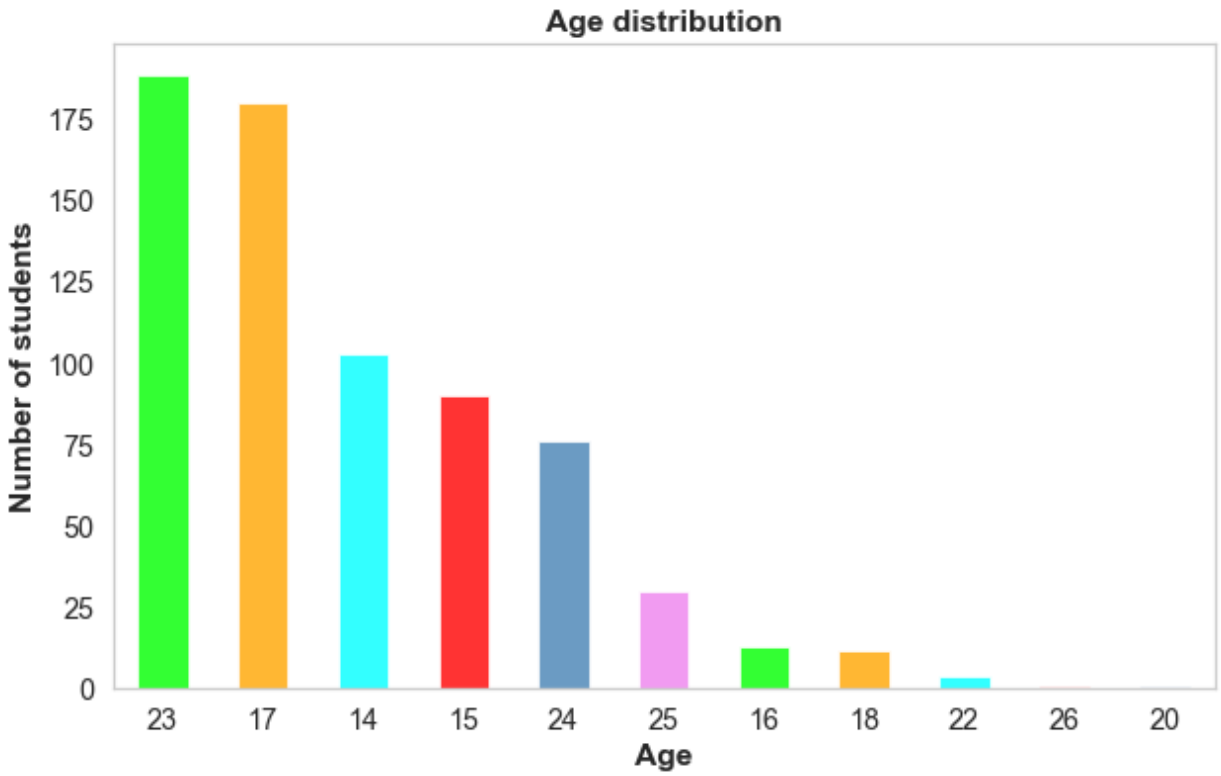
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Gender'], title='Gender distribution', xlabe

plt.show()
```



Plotting 'Age'

Let's check the histogram.

In [325…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Age'],
                     color=['lime', 'orange', 'cyan', 'red', 'steelblue', 'vi
                     title='Age distribution', xlabel='Age')

plt.show()
```



Plotting Frequently Visited Website'

Let's check the histogram.

In [326…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Frequently Visited Website'], rot=45,
                     color=['salmon', 'royalblue', 'violet', 'tomato', 'steelb
                     title='Frequently visited websites', xlabel='Website name

plt.show()
```

Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [327…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Frequently Visited Website',
                                 combined_df['Frequently Visited Website'].value_

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.125), dictionary['Google'], width/2, label = '(
rects2 = ax.bar(x - width, dictionary['Facebook'], width/2, label = 'Facebook
rects3 = ax.bar(x - width/2, dictionary['Youtube'], width/2, label = 'Youtube
rects4 = ax.bar(x, dictionary['Whatsapp'], width/2, label = 'Whatsapp')
rects5 = ax.bar(x + width/2, dictionary['Gmail'], width/2, label = 'Gmail')
rects6 = ax.bar(x + width, dictionary['Twitter'], width/2, label = 'Twitter')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Frequently Visited Websites vs Academic Performance', fontweig
ax.set_xticks(x - width/3)
ax.set_xticklabels(labels)
ax.legend(title='Frequently visited websites', title_fontsize=14, loc = 'upper

sns.set(font_scale=0.75)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Frequently_Visited_Websites_WRT_Academic_Performance_Frequency_Distri
plt.show()
```

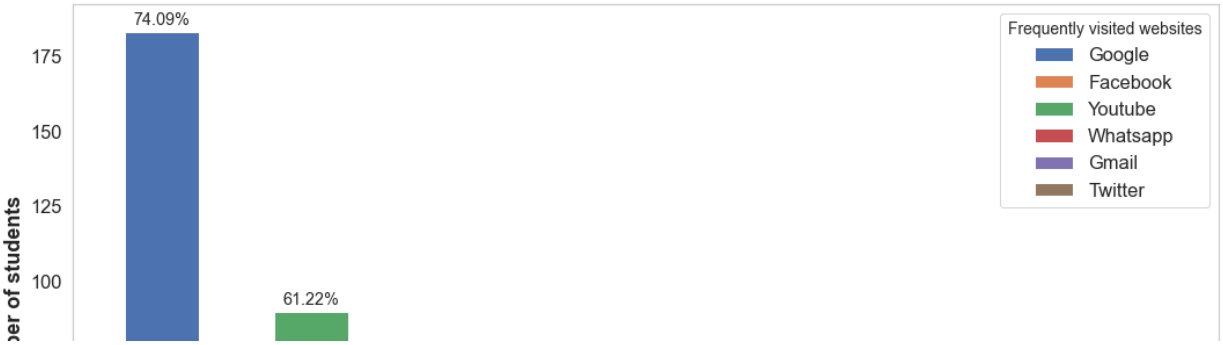
```
Saving figure Frequently_Visited_Websites_WRT_Academic_Performance_Frequency_D
istribution
```

Let's check the distribution of this feature against `'Browsing Purpose'`.

In [328…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_browsing_purpose(combined_df, 'Frequently Vis
                                combined_df['Frequently Visited Website'].value_

labels = ['Academic', 'Non-academic']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.125), dictionary['Google'], width/2, label = '(
rects2 = ax.bar(x - width, dictionary['Facebook'], width/2, label = 'Facebook
rects3 = ax.bar(x - width/2, dictionary['Youtube'], width/2, label = 'Youtube
rects4 = ax.bar(x, dictionary['Whatsapp'], width/2, label = 'Whatsapp')
rects5 = ax.bar(x + width/2, dictionary['Gmail'], width/2, label = 'Gmail')
rects6 = ax.bar(x + width, dictionary['Twitter'], width/2, label = 'Twitter')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Browsing purpose', fontweight = 'bold')
# ax.set_title('Frequently Visited Websites vs Browsing Purpose', fontweight =
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Frequently visited websites', title_fontsize=14, loc='upper r

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Frequently_Visited_Websites_WRT_Browsing_Purpose_Frequency_Distribut
plt.show()
```

```
Saving figure Frequently_Visited_Websites_WRT_Browsing_Purpose_Frequency_Distr
ibution
```

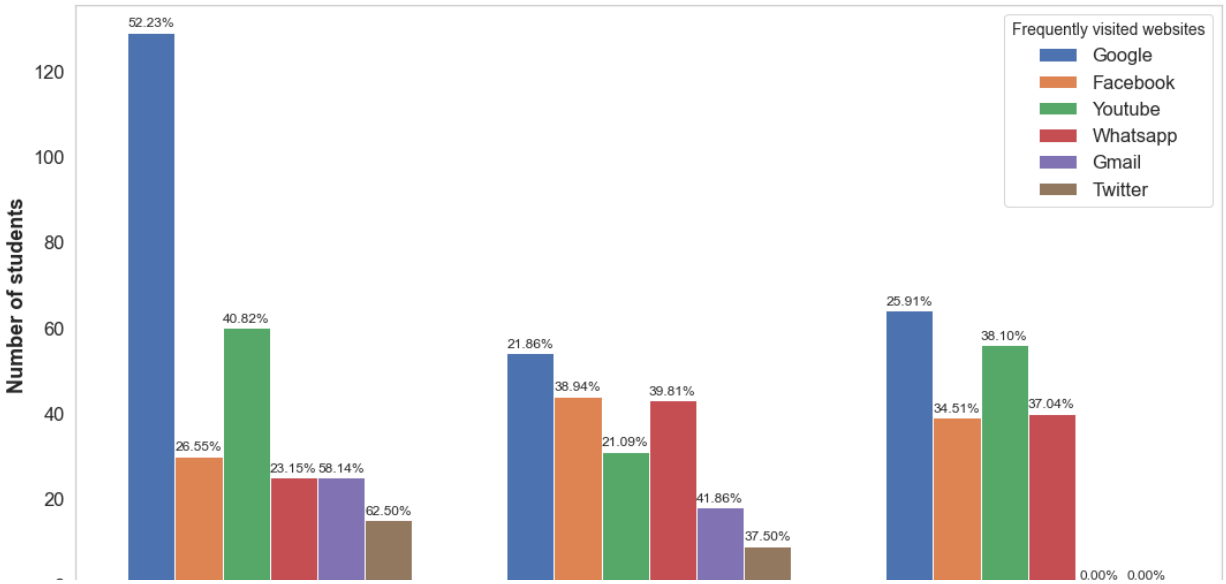


Let's check the distribution of this feature against `'Academic Institution'`.

In [329…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Frequently
                                combined_df['Frequently Visited Website'].value_

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.125), dictionary['Google'], width/2, label = '(
rects2 = ax.bar(x - width, dictionary['Facebook'], width/2, label = 'Facebook
rects3 = ax.bar(x - width/2, dictionary['Youtube'], width/2, label = 'Youtube
rects4 = ax.bar(x, dictionary['Whatsapp'], width/2, label = 'Whatsapp')
rects5 = ax.bar(x + width/2, dictionary['Gmail'], width/2, label = 'Gmail')
rects6 = ax.bar(x + width, dictionary['Twitter'], width/2, label = 'Twitter')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Frequently Visited Websites vs Academic Institution', fontwei(
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Frequently visited websites', title_fontsize=14, loc='upper r

sns.set(font_scale=1)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Frequently_Visited_Websites_WRT_Academic_Institution_Frequency_Distr

plt.show()
```

```
Saving figure Frequently_Visited_Websites_WRT_Academic_Institution_Frequency_D
istribution
```

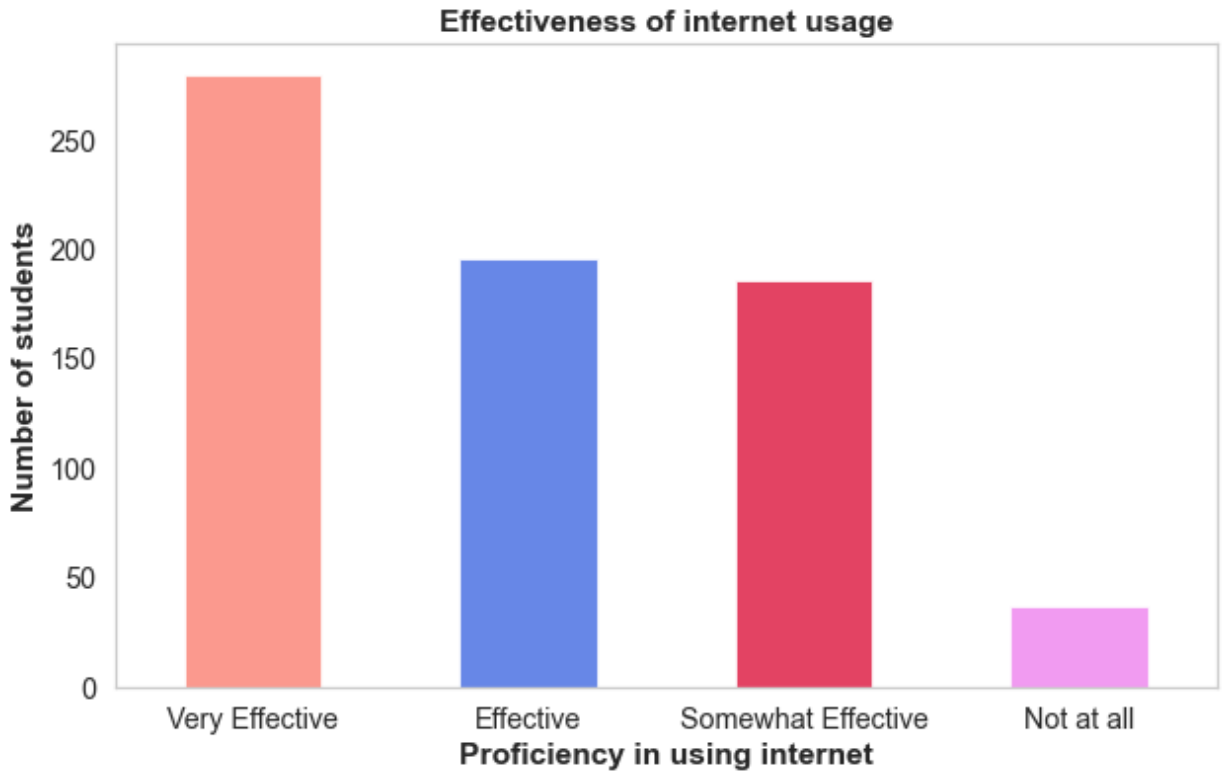


Plotting 'Effectiveness Of Internet Usage'

Let's check the histogram.

In [330…

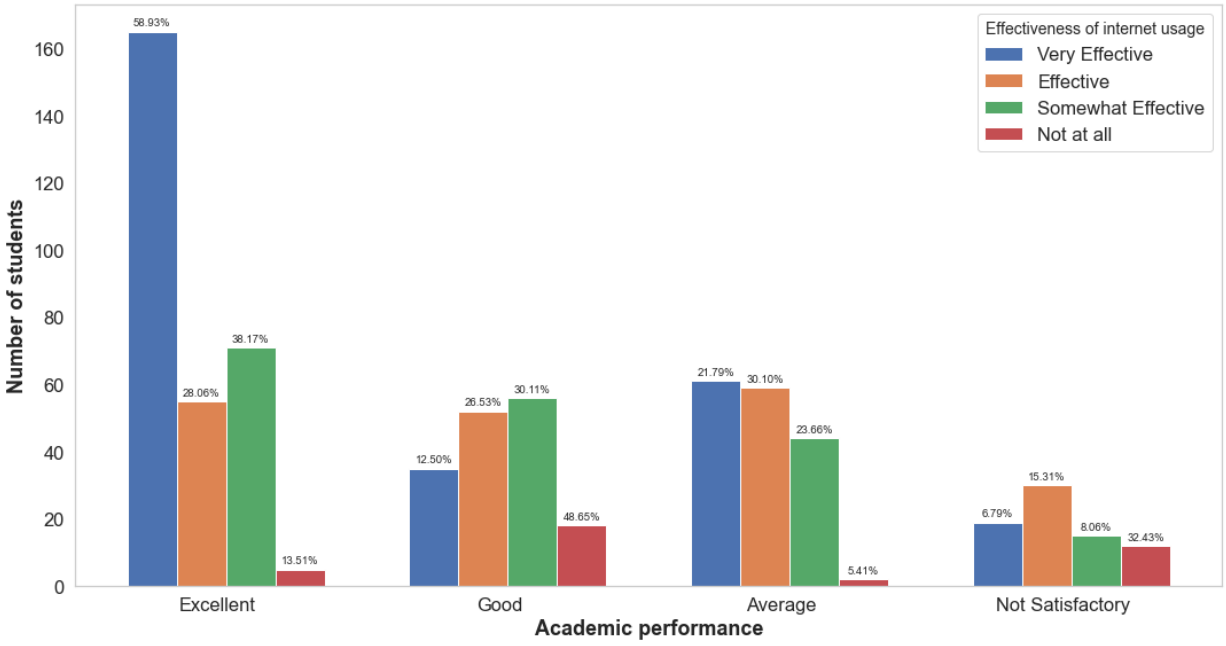
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Effectiveness Of Internet Usage'],
                     color=['salmon', 'royalblue', 'crimson', 'violet'],
                     title='Effectiveness of internet usage', xlabel='Proficie

plt.show()
```



Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [331…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Effectiveness Of Internet Usage
                                 ['Very Effective', 'Effective', 'Somewhat Effect

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.35

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Very Effective'], width/2, label = 'Ve
rects2 = ax.bar(x - width/2, dictionary['Effective'], width/2, label = 'Effect
rects3 = ax.bar(x, dictionary['Somewhat Effective'], width/2, label = 'Somewha
rects4 = ax.bar(x + width/2, dictionary['Not at all'], width/2, label = 'Not a

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Effectiveness Of Internet Usage vs Academic Performance', font
ax.set_xticks(x - width/3)
ax.set_xticklabels(labels)
ax.legend(title='Effectiveness of internet usage', title_fontsize=14)

sns.set(font_scale=0.8)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)

fig.tight_layout()

plt.show()
```

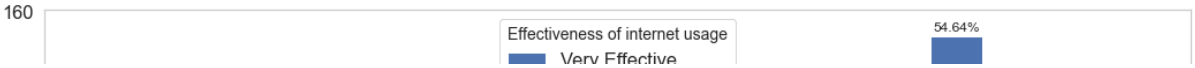


Let's check the distribution of this feature against `'Academic Institution'`.

In [332…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Effectiver
                                combined_df['Effectiveness Of Internet Usage'].v

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Very Effective'], width/2, label = 'Ver
rects2 = ax.bar(x - width/2, dictionary['Effective'], width/2, label = 'Effect
rects3 = ax.bar(x, dictionary['Somewhat Effective'], width/2, label = 'Somewha
rects4 = ax.bar(x + width/2, dictionary['Not at all'], width/2, label = 'Not a

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Frequently Visited Websites vs Academic Institution', fontweig
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Effectiveness of internet usage', title_fontsize=14, loc='upp

sns.set(font_scale=1)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)

fig.tight_layout()

save_fig('Effectiveness_Of_Internet_Usage_WRT_Academic_Institution_Frequency_D
plt.show()
```

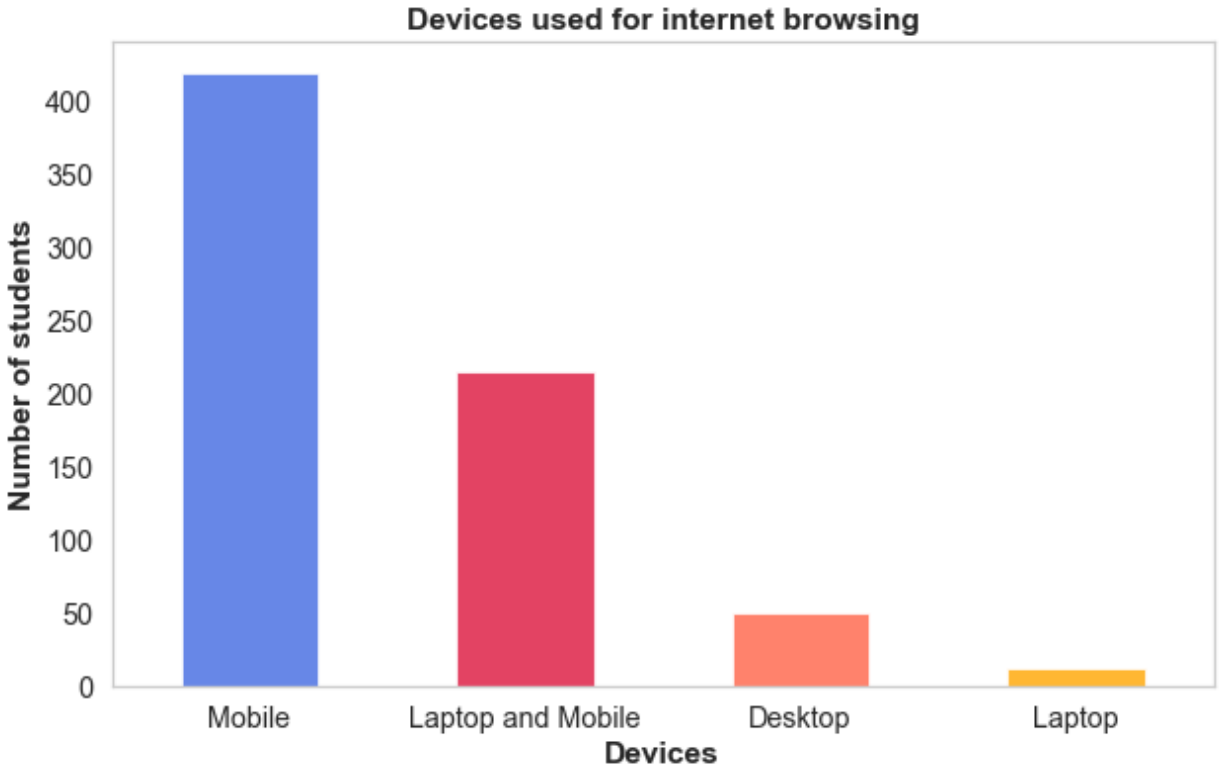
```
Saving figure Effectiveness_Of_Internet_Usage_WRT_Academic_Institution_Frequen
cy_Distribution
```



Plotting 'Devices Used For Internet Browsing'

Let's check the histogram.

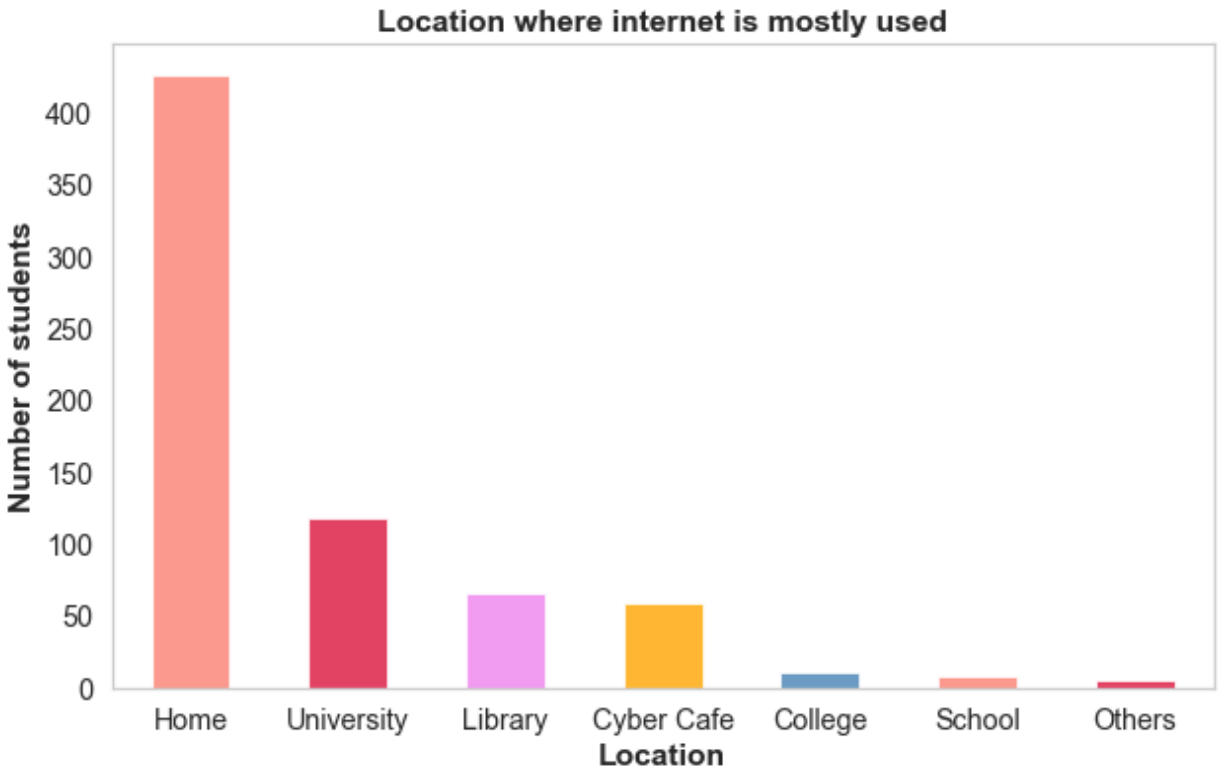
In [333…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Devices Used For Internet Browsing'],
                     color=['royalblue', 'crimson', 'tomato', 'orange'],
                     title='Devices used for internet browsing', xlabel='Devi

plt.show()
```



Plotting 'Location Of Internet Use'

Let's check the histogram.

In [334…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Location Of Internet Use'],
                     color=['salmon', 'crimson', 'violet', 'orange', 'steelblu
                     title='Location where internet is mostly used', xlabel='I

plt.show()
```



Let's check the distribution of this feature against `'Academic Institution'`.

In [335…

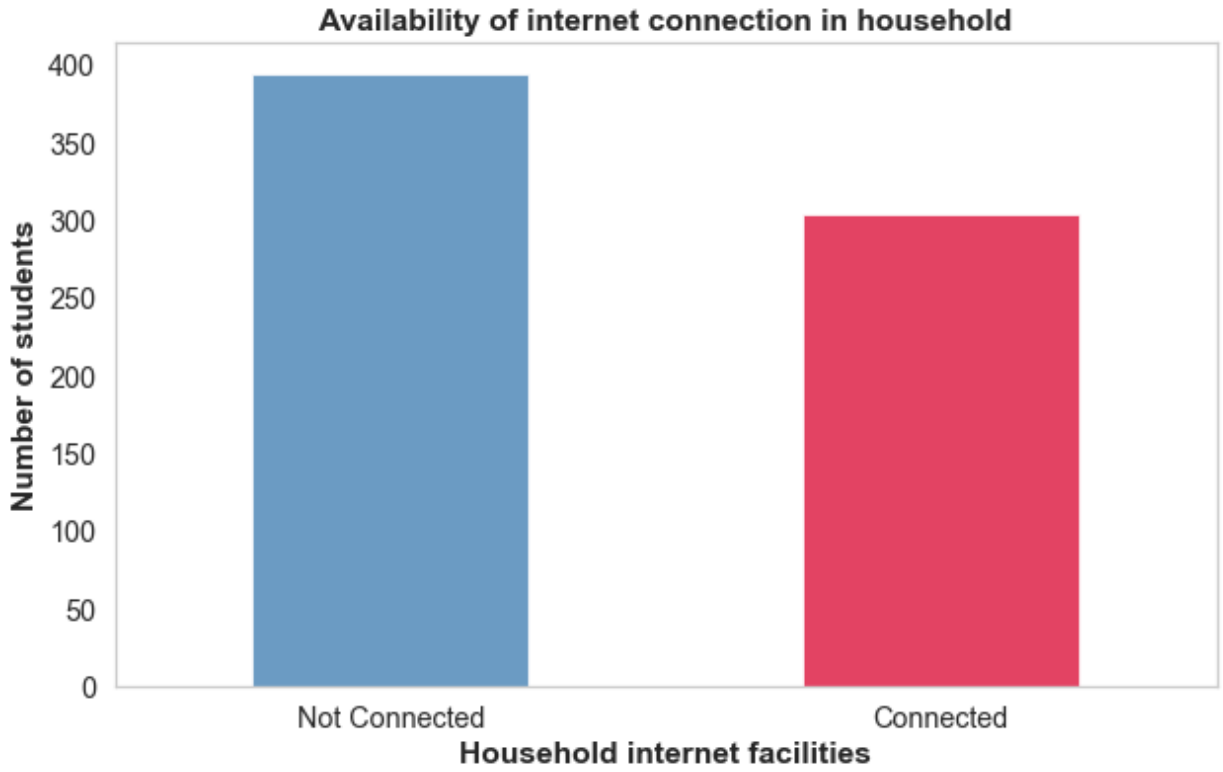
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Location (
                                combined_df['Location Of Internet Use'].value_co

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.12), dictionary['Home'], width/2, label = 'Home
rects2 = ax.bar(x - width, dictionary['University'], width/2, label = 'Univers
rects3 = ax.bar(x - width/2, dictionary['Library'], width/2, label = 'Library
rects4 = ax.bar(x, dictionary['Cyber Cafe'], width/2, label = 'Cyber Cafe')
rects5 = ax.bar(x + width/2, dictionary['College'], width/2, label = 'College
rects6 = ax.bar(x + width, dictionary['School'], width/2, label = 'School')
rects7 = ax.bar(x + (width + 0.125), dictionary['Others'], width/2, label = '(

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Location Of Internet Use vs Academic Institution', fontweight
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Location of internet use', title_fontsize=14, loc='upper righ

sns.set(font_scale=0.65)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)
autolabel(rects7)

fig.tight_layout()

save_fig('Location_Of_Internet_Use_WRT_Academic_Institution_Frequency_Distribu
plt.show()
```

```
Saving figure Location_Of_Internet_Use_WRT_Academic_Institution_Frequency_Dist
ribution
```

Plotting 'Household Internet Facilities'

In [336…

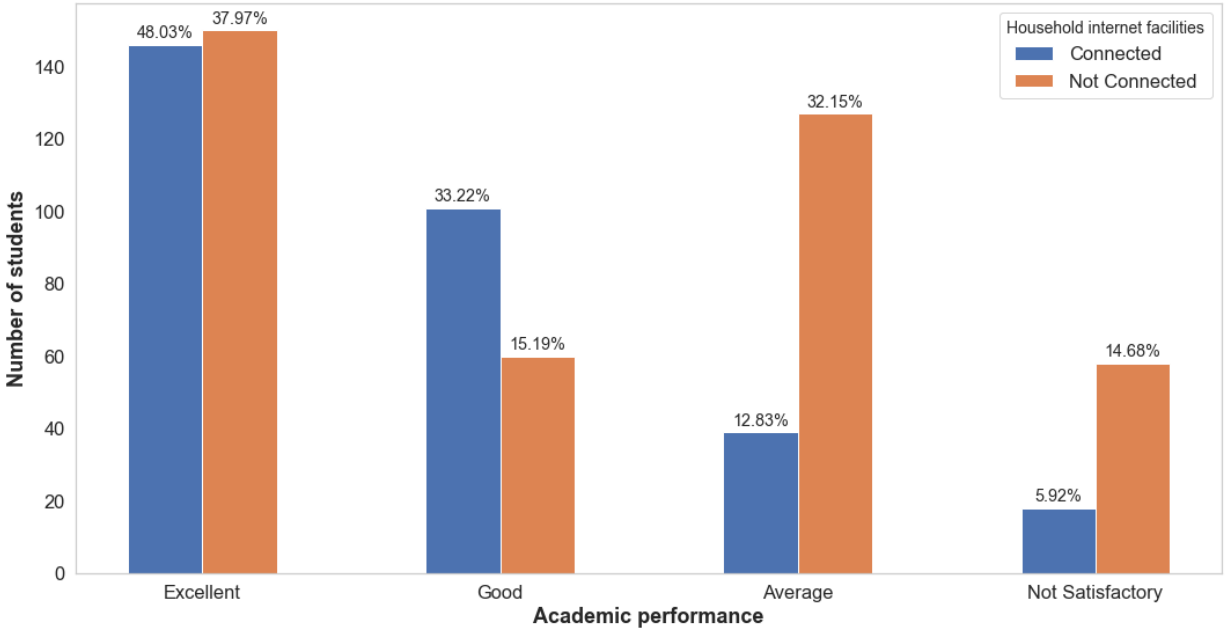
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Household Internet Facilities'],
                     title='Availability of internet connection in household',
                     xlabel='Household internet facilities')

plt.show()
```



Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [337…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Household Internet Facilities',
                                 combined_df['Household Internet Facilities'].val

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Connected'], width, label = 'Connected
rects2 = ax.bar(x, dictionary['Not Connected'], width, label = 'Not Connected

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Availability Of Internet Connection In Household vs Academic
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Household internet facilities', title_fontsize=14)

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```

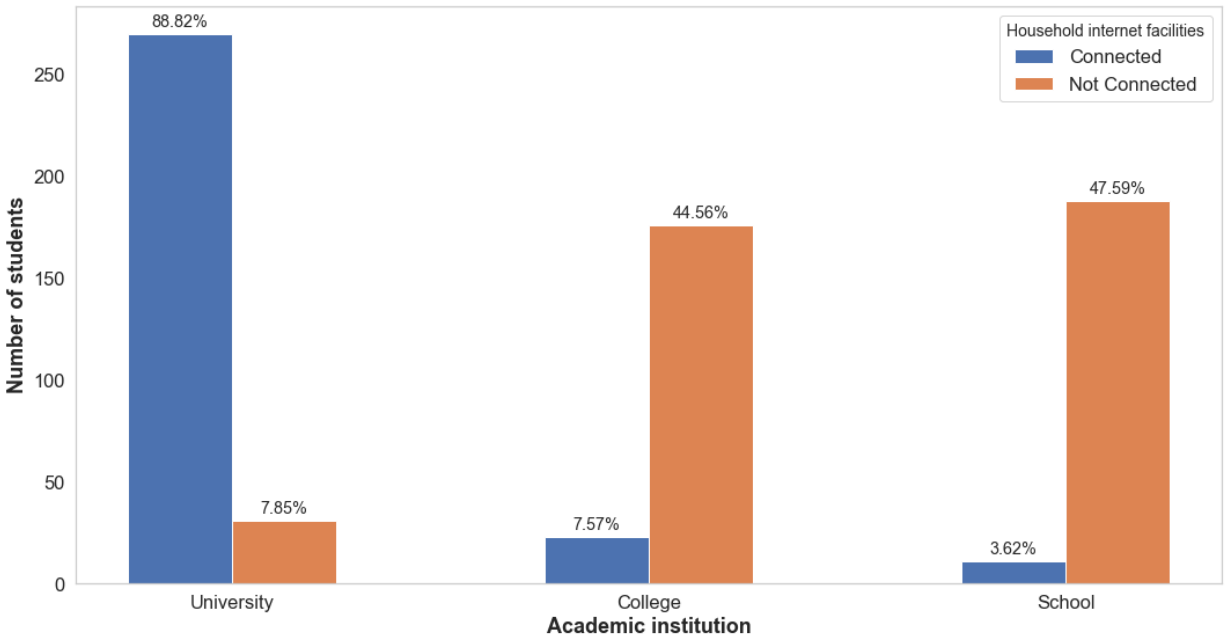


Let's check the distribution of this feature against `'Academic Institution'`.

In [338…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Household
                                combined_df['Household Internet Facilities'].va

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Connected'], width, label = 'Connected
rects2 = ax.bar(x, dictionary['Not Connected'], width, label = 'Not Connected

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Availability Of Internet Connection In Household vs Academic
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Household internet facilities', title_fontsize=14, loc='uppe

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

save_fig('Household_Internet_Facilities_WRT_Academic_Institution_Frequency_Dis
plt.show()
```

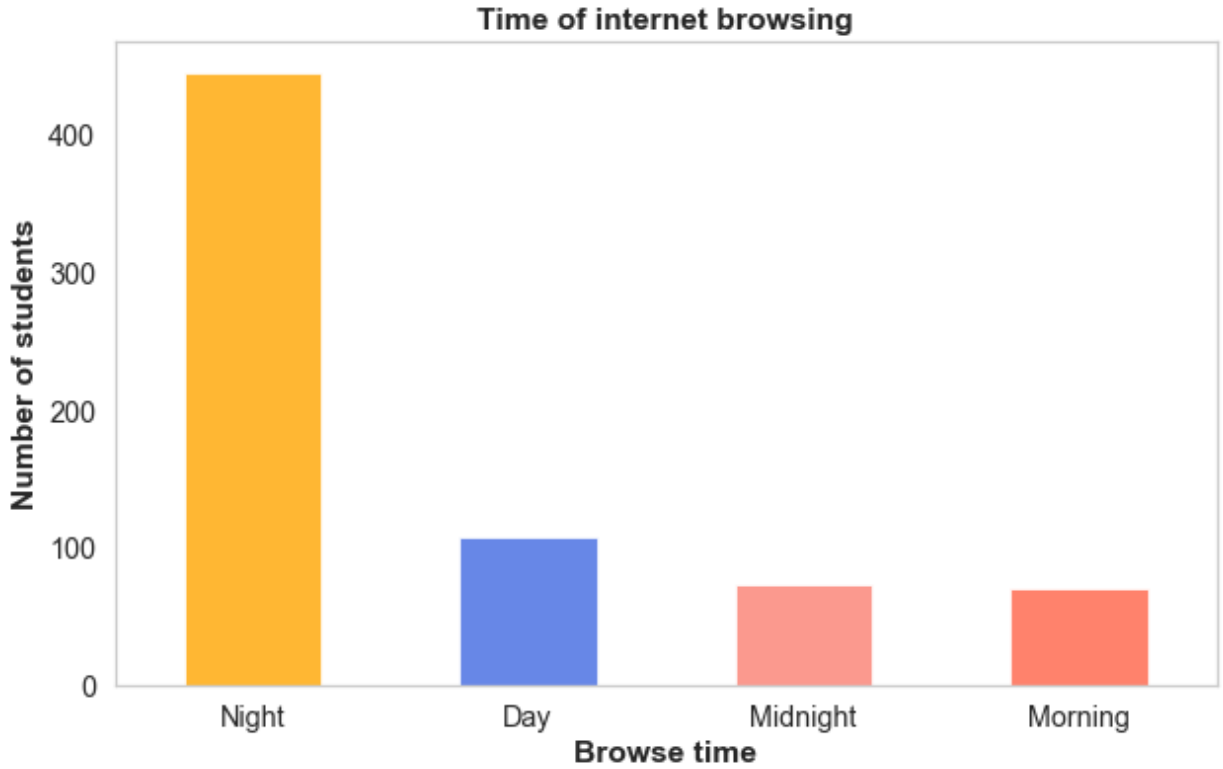
```
Saving figure Household_Internet_Facilities_WRT_Academic_Institution_Frequency
_Distribution
```



Plotting 'Time Of Internet Browsing'

Let's check the histogram.

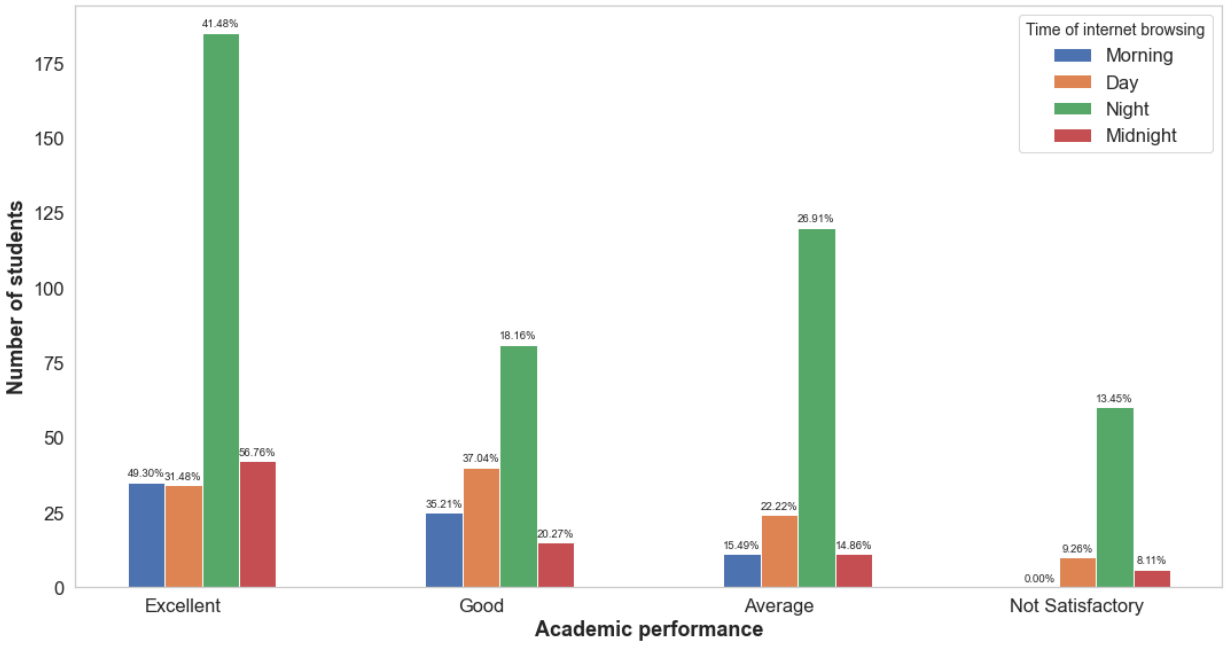
In [339…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Time Of Internet Browsing'], color=['orange
                     title='Time of internet browsing', xlabel='Browse time')

plt.show()
```



Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [340…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Time Of Internet Browsing',
                                 combined_df['Time Of Internet Browsing'].value_

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Morning'], width/2, label = 'Morning')
rects2 = ax.bar(x - width/2, dictionary['Day'], width/2, label = 'Day')
rects3 = ax.bar(x, dictionary['Night'], width/2, label = 'Night')
rects4 = ax.bar(x + width/2, dictionary['Midnight'], width/2, label = 'Midnigh

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Time Of Internet Browsing vs Academic Performance', fontweigh
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Time of internet browsing', title_fontsize=14)

sns.set(font_scale=0.8)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)

fig.tight_layout()

plt.show()
```



Let's check the distribution of this feature against `'Academic Institution'`.

In [341…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Time Of In
                                combined_df['Time Of Internet Browsing'].value

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Morning'], width/2, label = 'Morning')
rects2 = ax.bar(x - width/2, dictionary['Day'], width/2, label = 'Day')
rects3 = ax.bar(x, dictionary['Night'], width/2, label = 'Night')
rects4 = ax.bar(x + width/2, dictionary['Midnight'], width/2, label = 'Midnigh

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Time Of Internet Browsing vs Academic Institution', fontweigh
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Time of internet browsing', title_fontsize=14)

sns.set(font_scale=0.95)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)

fig.tight_layout()

save_fig('Time_Of_Internet_Browsing_WRT_Academic_Institution_Frequency_Distrib

plt.show()
```

```
Saving figure Time_Of_Internet_Browsing_WRT_Academic_Institution_Frequency_Dis
tribution
```

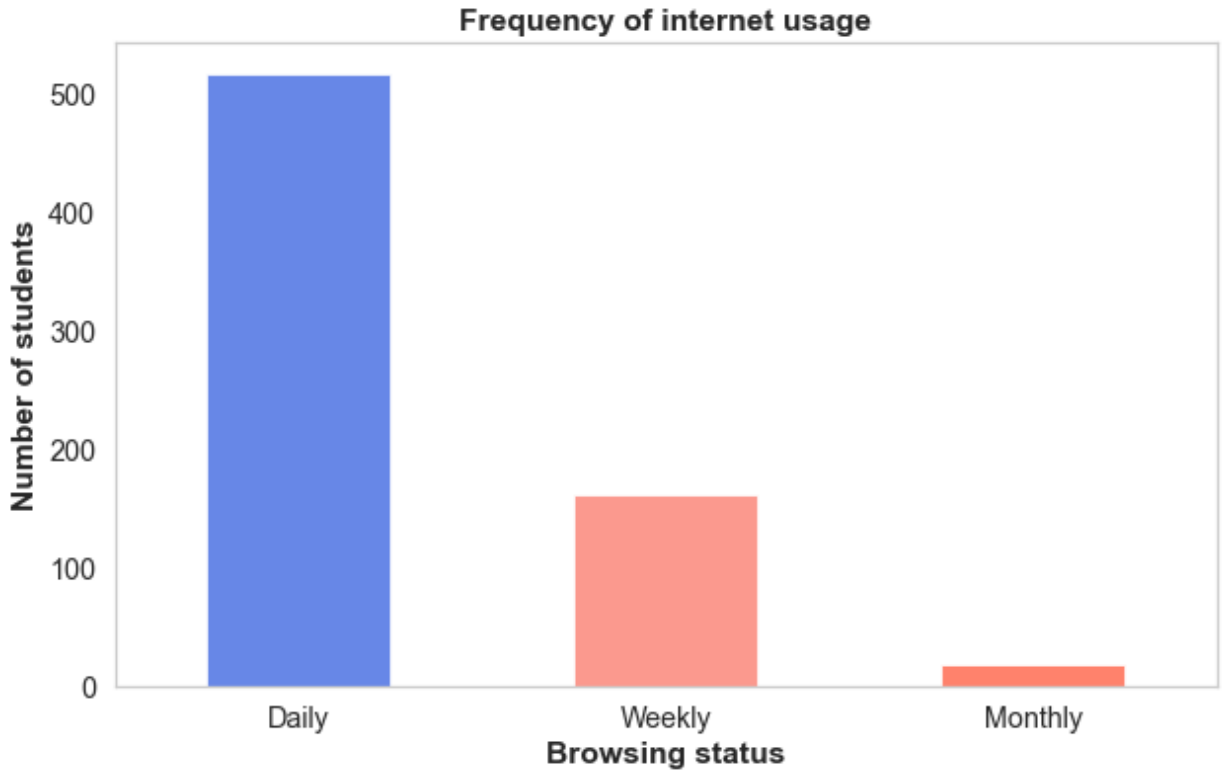
175 Time of internet browsing 37.67% 38.57%

Plotting 'Frequency Of Internet Usage'

Let's check the histogram.

In [342…

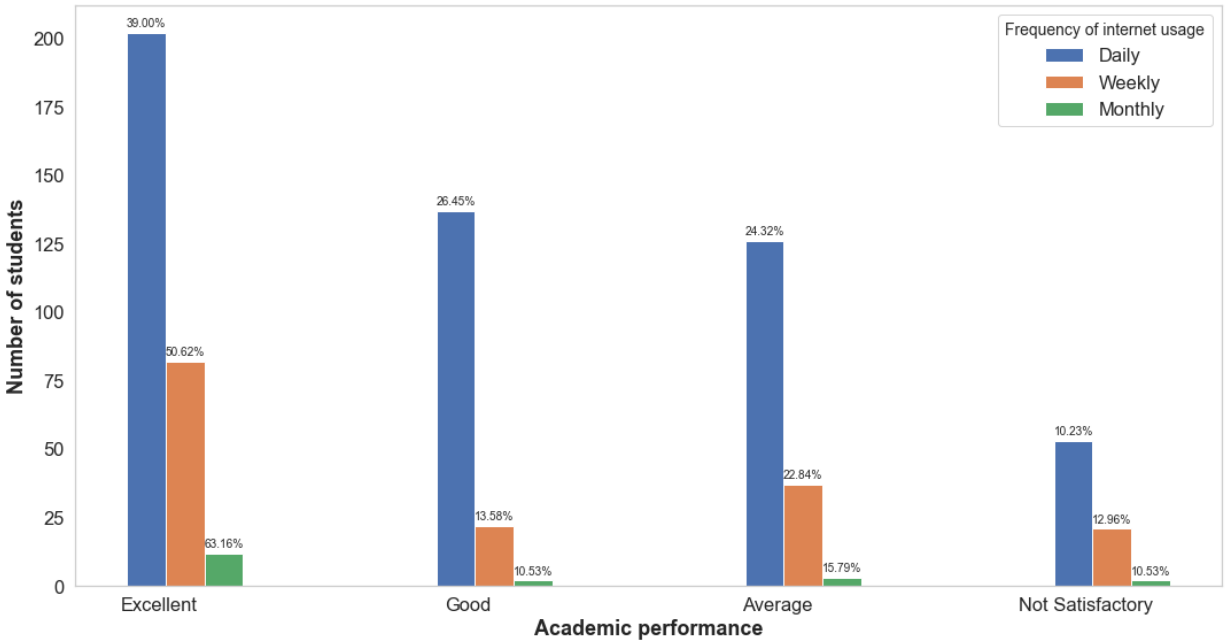
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Frequency Of Internet Usage'], color=['royal
                     title='Frequency of internet usage', xlabel='Browsing sta

plt.show()
```



Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [343…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Frequency Of Internet Usage',
                                 ['Daily', 'Weekly', 'Monthly'])

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width/2, dictionary['Daily'], width/2, label = 'Daily')
rects2 = ax.bar(x, dictionary['Weekly'], width/2, label = 'Weekly')
rects3 = ax.bar(x + width/2, dictionary['Monthly'], width/2, label = 'Monthly

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Frequency Of Internet Usage vs Academic Performance', fontwei
ax.set_xticks(x - width/3)
ax.set_xticklabels(labels)
ax.legend(title='Frequency of internet usage', title_fontsize=14)

sns.set(font_scale=0.85)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

fig.tight_layout()

plt.show()
```

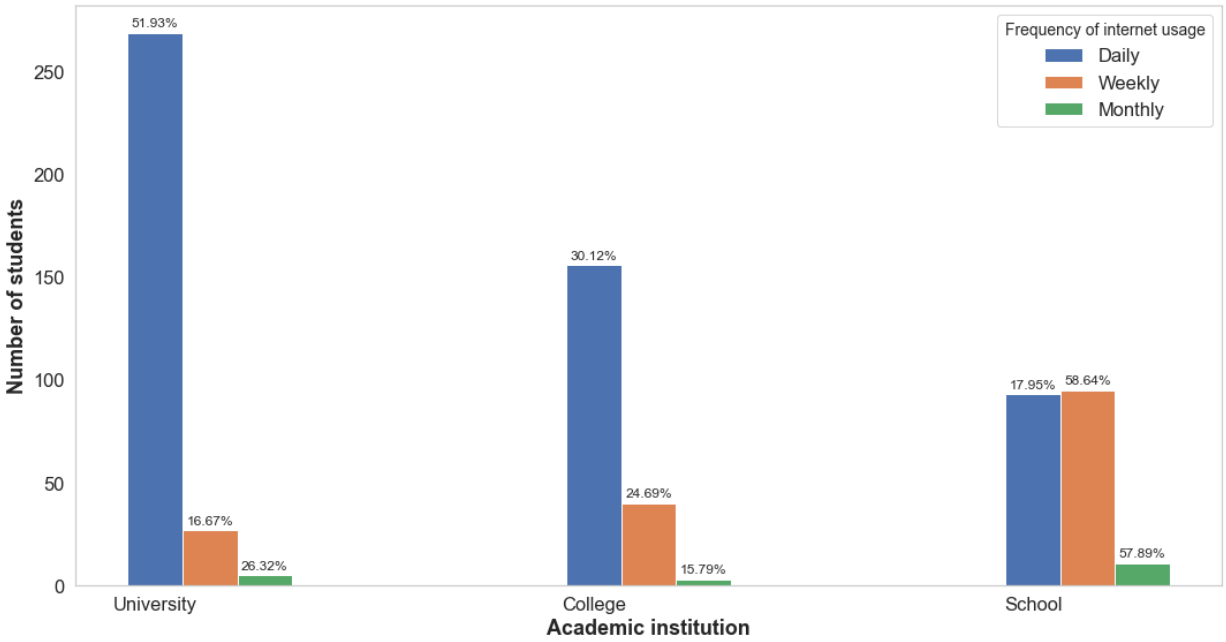


Let's check the distribution of this feature against `'Academic Institution'`.

In [344…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Frequency
                                combined_df['Frequency Of Internet Usage'].va

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width/2, dictionary['Daily'], width/2, label = 'Daily')
rects2 = ax.bar(x, dictionary['Weekly'], width/2, label = 'Weekly')
rects3 = ax.bar(x + width/2, dictionary['Monthly'], width/2, label = 'Monthly

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Frequency Of Internet Usage vs Academic Institution', fontwei
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Frequency of internet usage', title_fontsize=14)

sns.set(font_scale=1.0)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

fig.tight_layout()

save_fig('Frequency_Of_Internet_Usage_WRT_Academic_Institution_Frequency_Dist
plt.show()
```

```
Saving figure Frequency_Of_Internet_Usage_WRT_Academic_Institution_Frequency_D
istribution
```



Plotting 'Place Of Student's Residence'

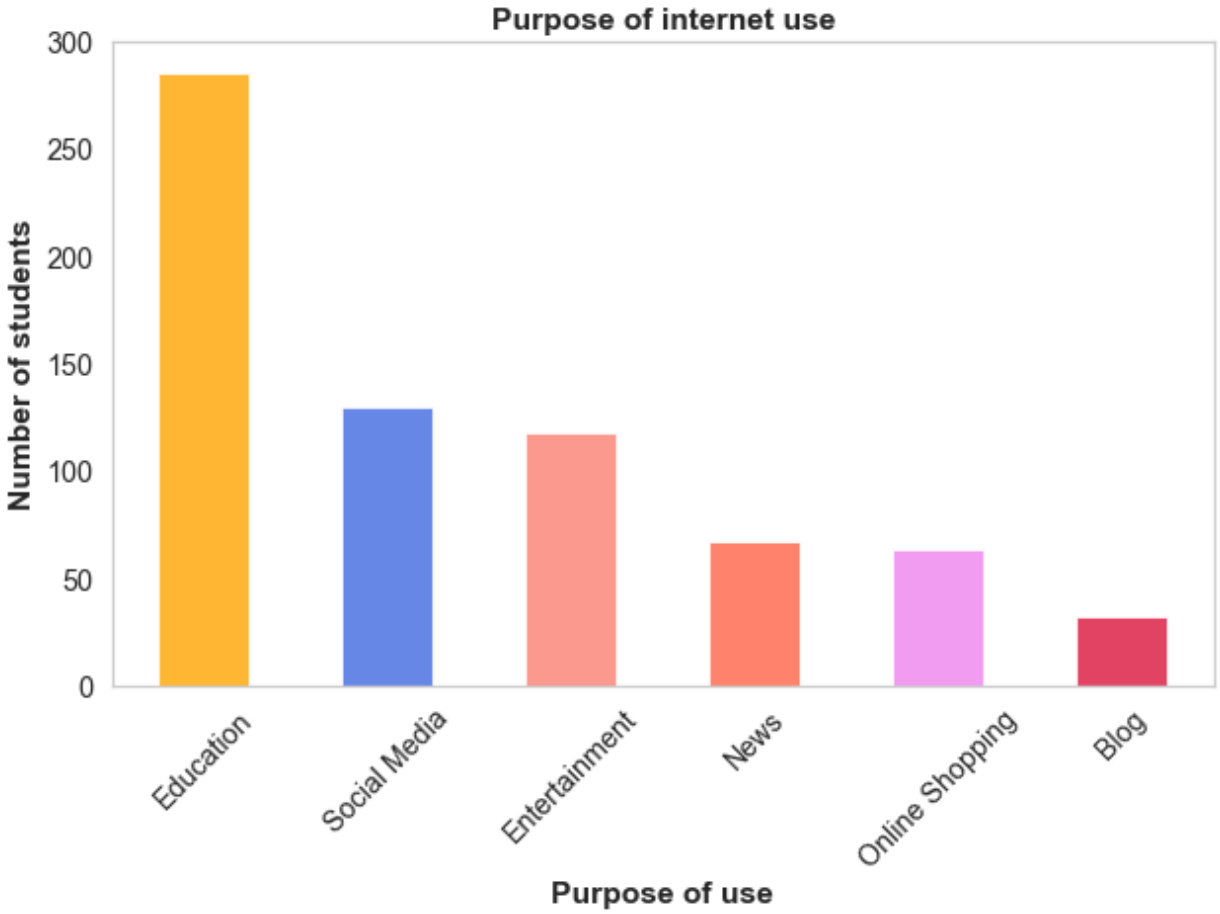
Let's check the histogram.

In [345…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Place Of Student\'s Residence'], color=['cri
                     title='Place of student\'s residence', xlabel='Location
                     

plt.show()
```

Plotting 'Purpose Of Internet Use'

Let's check the histogram.

In [346…
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Purpose Of Internet Use'], rot=45,
                     color = ['orange', 'royalblue', 'salmon', 'tomato', 'viol
                     title='Purpose of internet use', xlabel='Purpose of use'

plt.show()
```



Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [347…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Purpose Of Internet Use',
                                 combined_df['Purpose Of Internet Use'].value_cou

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.125), dictionary['Social Media'], width/2, labe
rects2 = ax.bar(x - width, dictionary['Education'], width/2, label = 'Educatio
rects3 = ax.bar(x - width/2, dictionary['Entertainment'], width/2, label = 'En
rects4 = ax.bar(x, dictionary['News'], width/2, label = 'News')
rects5 = ax.bar(x + width/2, dictionary['Online Shopping'], width/2, label =
rects6 = ax.bar(x + width, dictionary['Blog'], width/2, label = 'Blog')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Purpose Of Internet Use vs Academic Performance', fontweight =
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Purpose of internet use', title_fontsize=14)

sns.set(font_scale=0.7)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Purpose_Of_Internet_Use_WRT_Academic_Performance_Frequency_Distribut
plt.show()
```

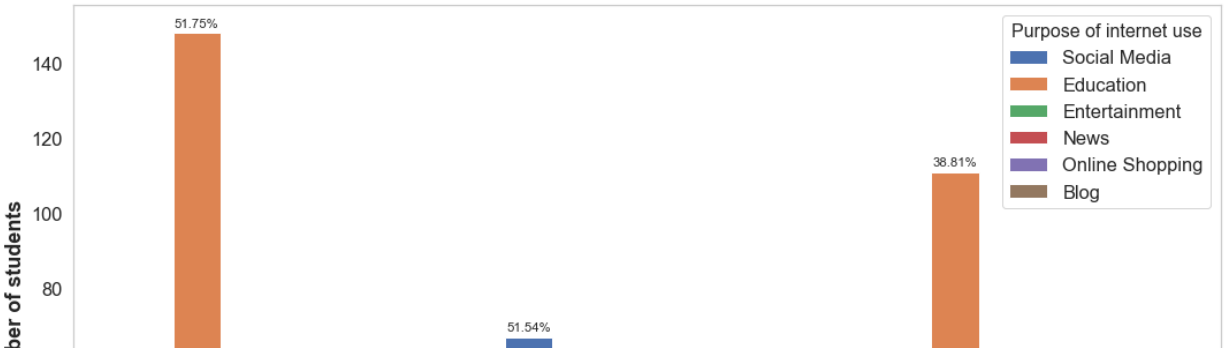
```
Saving figure Purpose_Of_Internet_Use_WRT_Academic_Performance_Frequency_Distr
ibution
```

Let's check the distribution of this feature against `'Academic Institution'`.

In [348…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Purpose Of
                                combined_df['Purpose Of Internet Use'].value_c

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.125), dictionary['Social Media'], width/2, labe
rects2 = ax.bar(x - width, dictionary['Education'], width/2, label = 'Educatio
rects3 = ax.bar(x - width/2, dictionary['Entertainment'], width/2, label = 'En
rects4 = ax.bar(x, dictionary['News'], width/2, label = 'News')
rects5 = ax.bar(x + width/2, dictionary['Online Shopping'], width/2, label =
rects6 = ax.bar(x + width, dictionary['Blog'], width/2, label = 'Blog')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Purpose Of Internet Use vs Academic Institution', fontweight =
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Purpose of internet use', title_fontsize=16)

sns.set(font_scale=0.95)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Purpose_Of_Internet_Use_WRT_Academic_Institution_Frequency_Distribut
plt.show()
```

```
Saving figure Purpose_Of_Internet_Use_WRT_Academic_Institution_Frequency_Distr
ibution
```

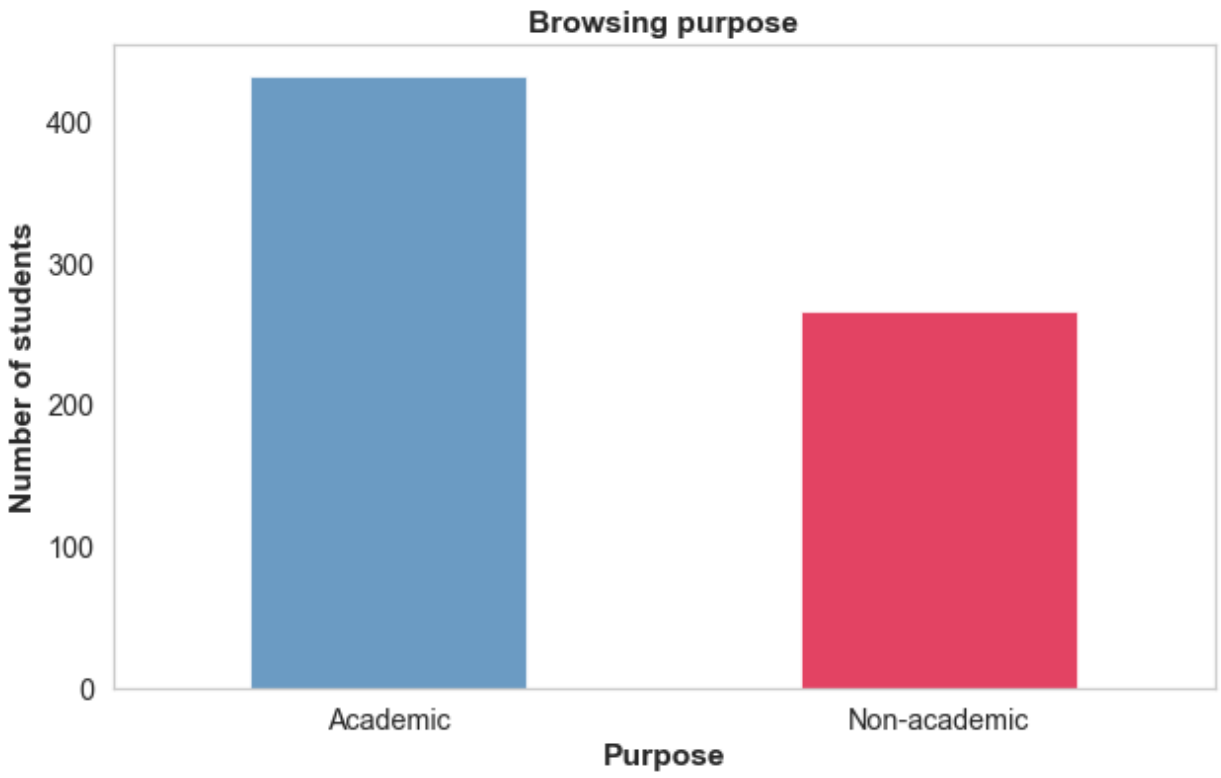


Plotting 'Browsing Purpose'

Let's check the histogram.

In [349…

```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Browsing Purpose'], title='Browsing purpose
                     xlabel='Purpose')

plt.show()
```

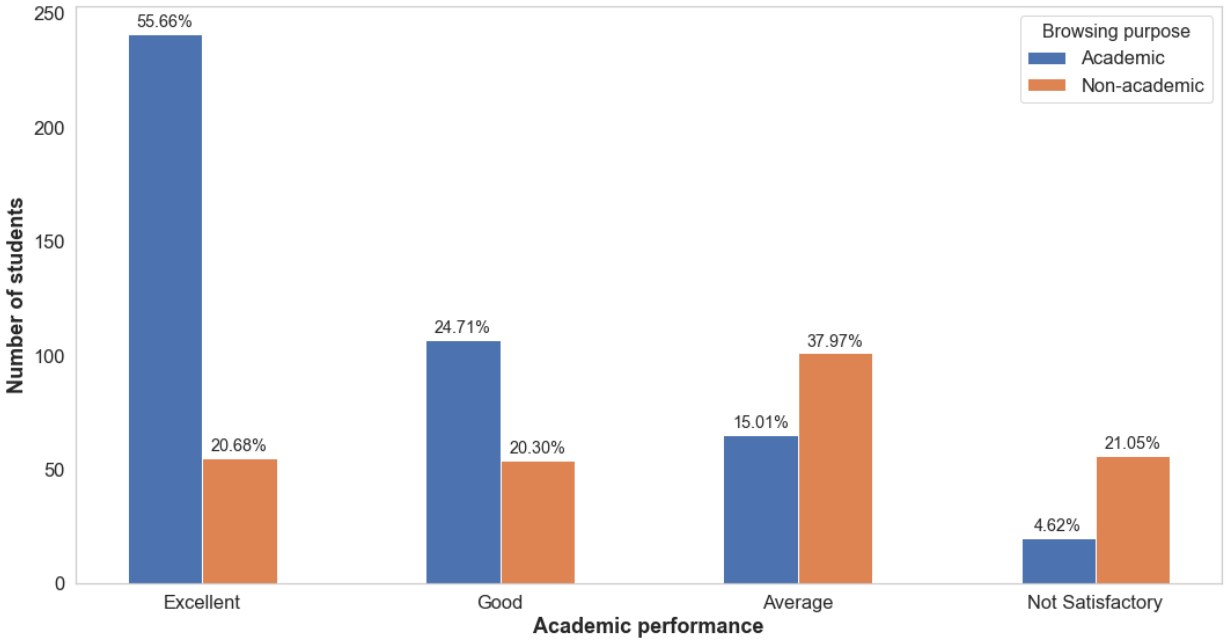


Let's check the distribution of this feature against the target i.e. 'Academic Performance' .

In [350…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Browsing Purpose',
                                 combined_df['Browsing Purpose'].value_counts().i

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Academic'], width, label = 'Academic')
rects2 = ax.bar(x, dictionary['Non-academic'], width, label = 'Non-academic')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Browsing Purpose vs Academic Performance', fontweight = 'bold
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Browsing purpose', title_fontsize=16, loc='upper right')

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

save_fig('Browsing_Purpose_WRT_Academic_Performance_Frequency_Distribution')
plt.show()
```

```
Saving figure Browsing_Purpose_WRT_Academic_Performance_Frequency_Distribution
```

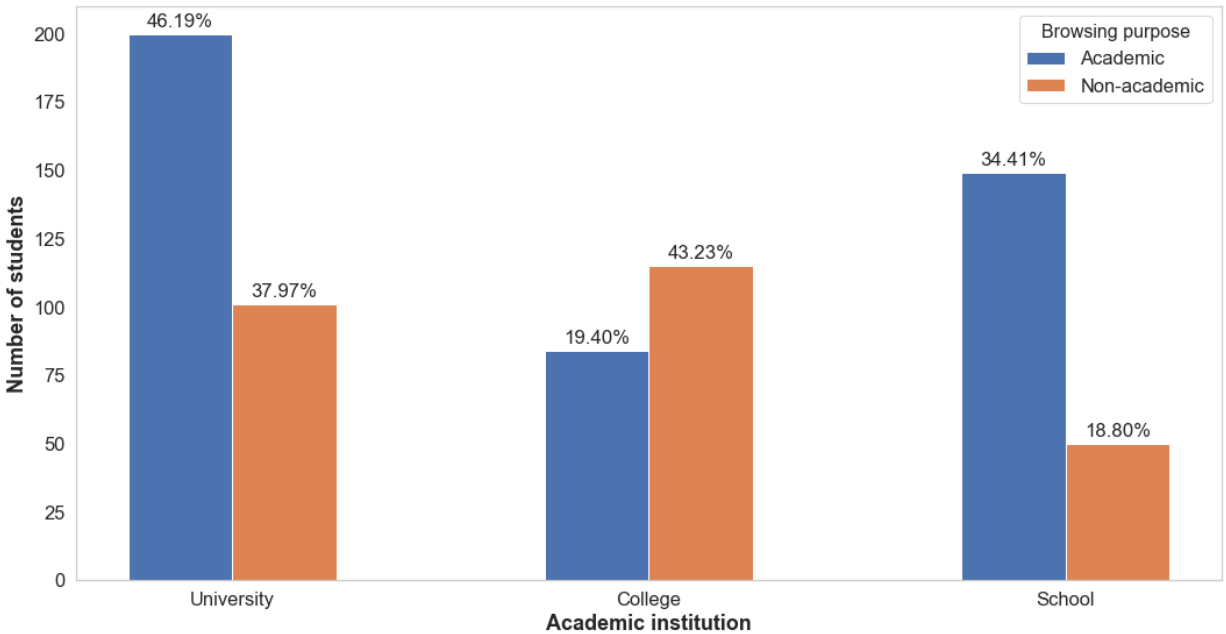


Let's check the distribution of this feature against `'Academic Institution'`.

In [351…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Browsing
                                combined_df['Browsing Purpose'].value_counts()

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Academic'], width, label = 'Academic')
rects2 = ax.bar(x, dictionary['Non-academic'], width, label = 'Non-academic')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Browsing Purpose vs Academic Institution', fontweight = 'bold
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Browsing purpose', title_fontsize=16, loc='upper right')

sns.set(font_scale=1.4)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

save_fig('Browsing_Purpose_WRT_Academic_Institution_Frequency_Distribution')
plt.show()
```

```
Saving figure Browsing_Purpose_WRT_Academic_Institution_Frequency_Distribution
```



Plotting 'Webinar'

Let's check the histogram.

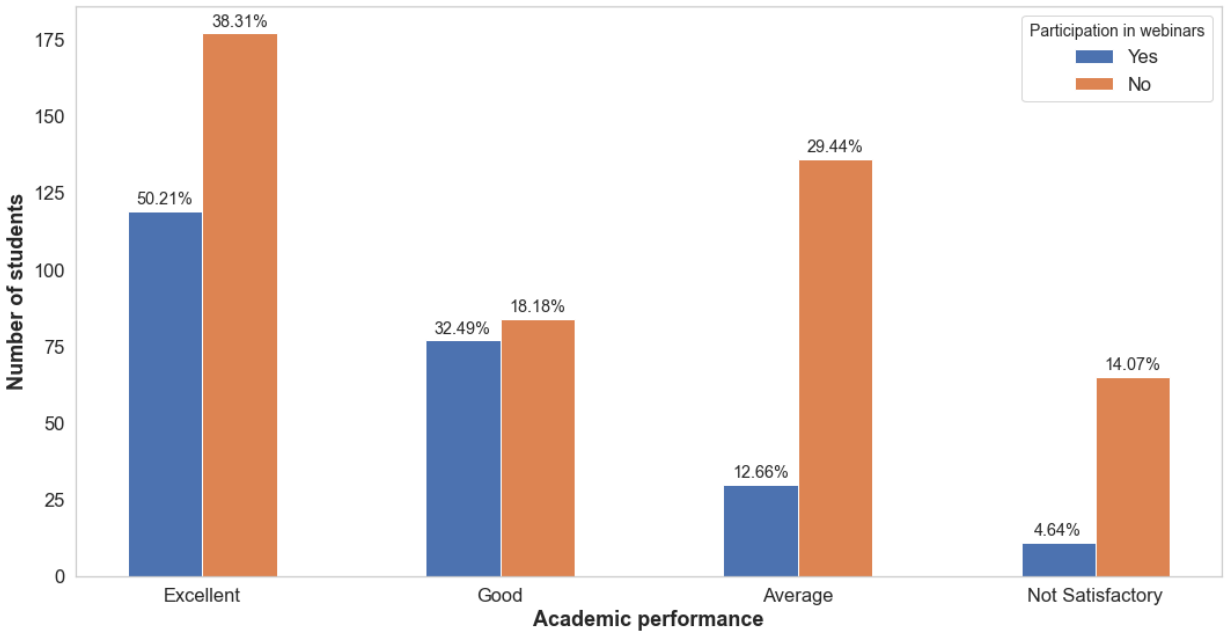
In [352…

```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Webinar'], color=['salmon', 'crimson'],
                     title='Participation in webinars', xlabel='Participation

plt.show()
```

Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [353…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Webinar',
                                 combined_df['Webinar'].value_counts().index.toli

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Yes'], width, label = 'Yes')
rects2 = ax.bar(x, dictionary['No'], width, label = 'No')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Participation In Webinars vs Academic Performance', fontweigh
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Participation in webinars', title_fontsize=14, loc='upper ri

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

plt.show()
```

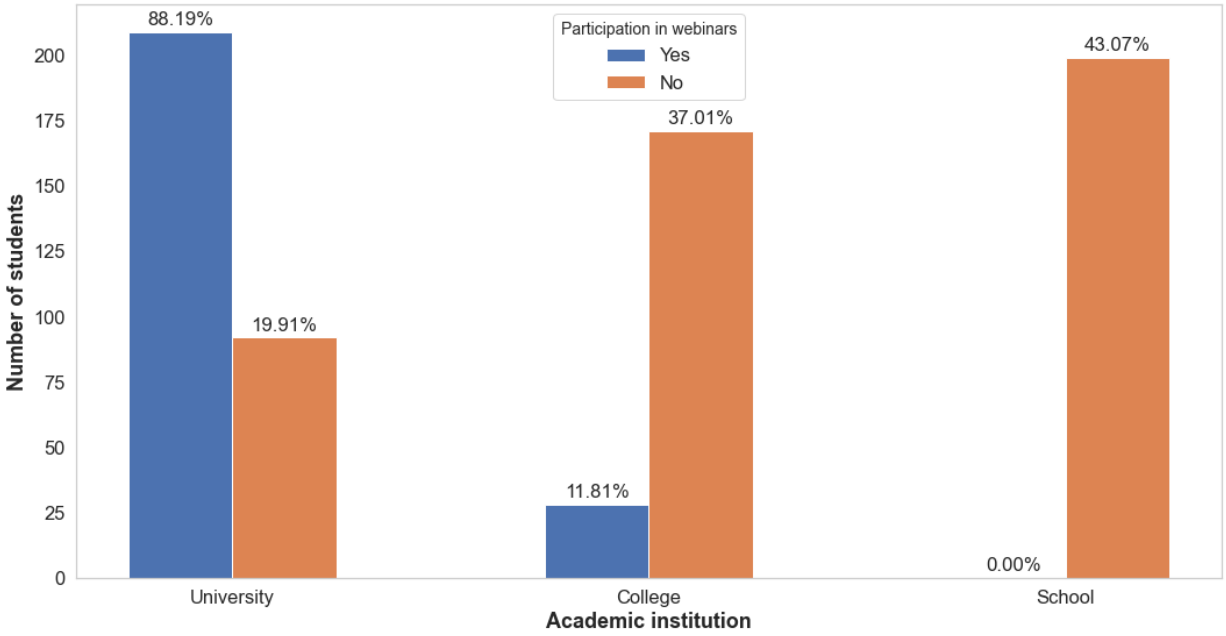


Let's check the distribution of this feature against `'Academic Institution'`.

In [354…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Webinar',
                                    combined_df['Webinar'].value_counts().index.to

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Yes'], width, label = 'Yes')
rects2 = ax.bar(x, dictionary['No'], width, label = 'No')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Webinar vs Academic Institution', fontweight = 'bold')
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Participation in webinars', title_fontsize=14, loc='upper cen

sns.set(font_scale=1.4)

autolabel(rects1)
autolabel(rects2)

fig.tight_layout()

save_fig('Webinar_WRT_Academic_Institution_Frequency_Distribution')
plt.show()
```

```
Saving figure Webinar_WRT_Academic_Institution_Frequency_Distribution
```

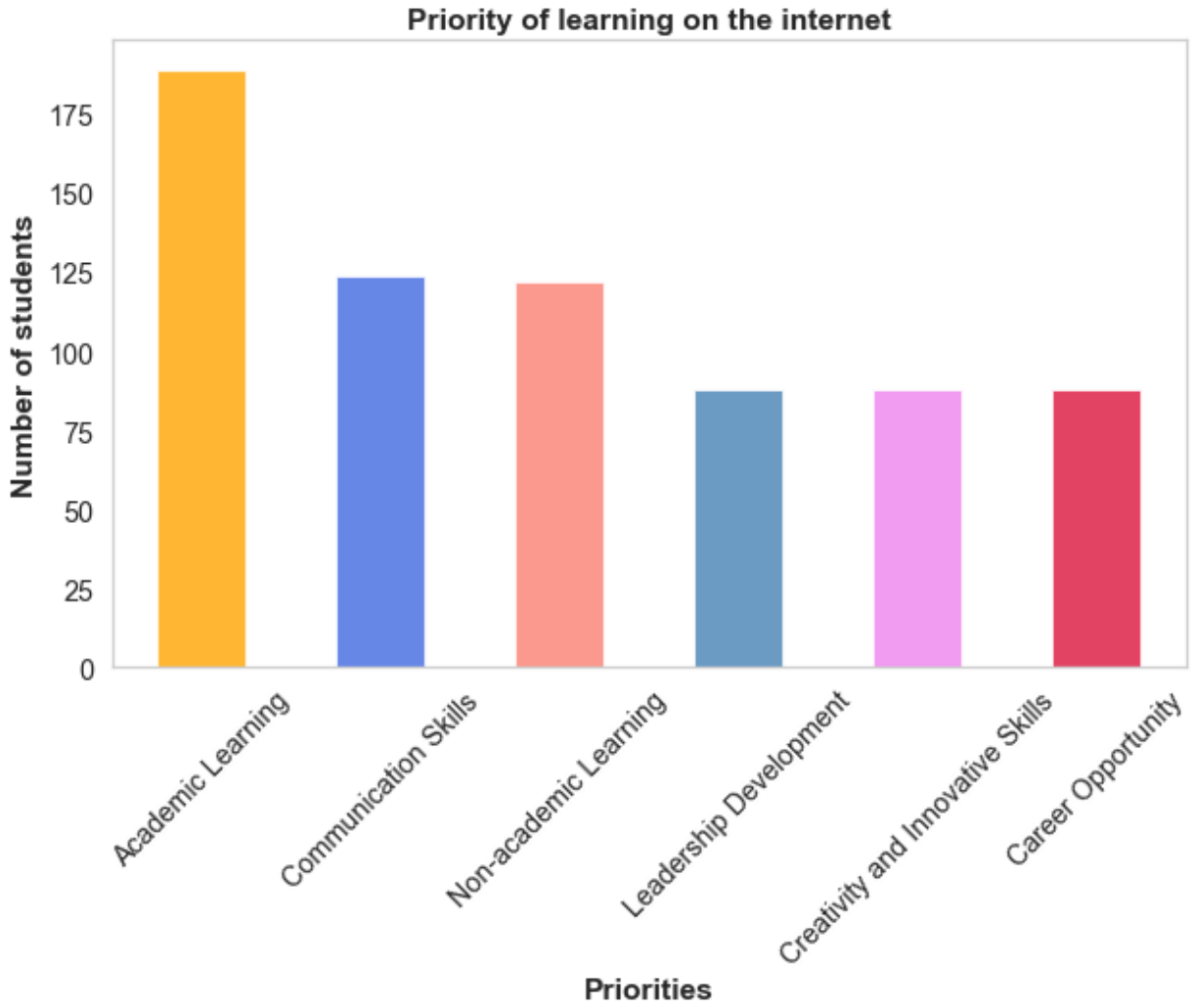


Plotting 'Priority Of Learning On The Internet'

Let's check the histogram.

In [355…

```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Priority Of Learning On The Internet'], rot=
                     color = ['orange', 'royalblue', 'salmon', 'steelblue', 'v
                     title='Priority of learning on the internet', xlabel='Pri

plt.show()
```

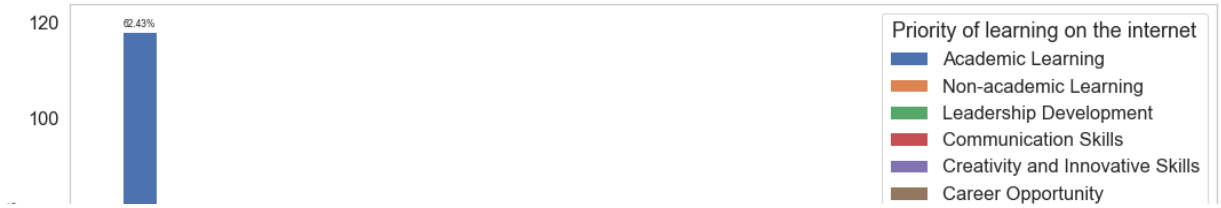


Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [356…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Priority Of Learning On The Int
                                 ['Academic Learning', 'Non-academic Learning',
                                  'Communication Skills', 'Creativity and Innovat

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.12), dictionary['Academic Learning'], width/2,
rects2 = ax.bar(x - width, dictionary['Non-academic Learning'], width/2, label
rects3 = ax.bar(x - width/2, dictionary['Leadership Development'], width/2, la
rects4 = ax.bar(x, dictionary['Communication Skills'], width/2, label = 'Commu
rects5 = ax.bar(x + width/2, dictionary['Creativity and Innovative Skills'], w
                label = 'Creativity and Innovative Skills')
rects6 = ax.bar(x + width, dictionary['Career Opportunity'], width/2, label =

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Priority Of Learning On The Internet vs Academic Performance'
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Priority of learning on the internet', title_fontsize=18, loc

sns.set(font_scale=0.7)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Priority_Of_Learning_On_The_Internet_W.R.T._Academic_Performance_Fre
plt.show()
```

```
Saving figure Priority_Of_Learning_On_The_Internet_W.R.T._Academic_Performance
_Frequency_Distribution
```

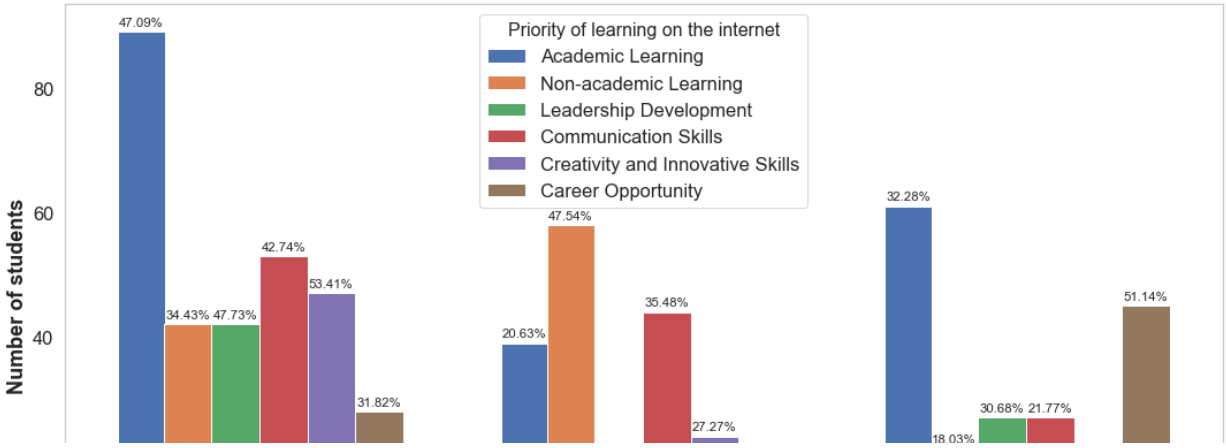


Let's check the distribution of this feature against `'Academic Institution'`.

In [357…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Priority
                                combined_df['Priority Of Learning On The Inte

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - (width + 0.12), dictionary['Academic Learning'], width/2,
rects2 = ax.bar(x - width, dictionary['Non-academic Learning'], width/2, label
rects3 = ax.bar(x - width/2, dictionary['Leadership Development'], width/2, la
rects4 = ax.bar(x, dictionary['Communication Skills'], width/2, label = 'Commu
rects5 = ax.bar(x + width/2, dictionary['Creativity and Innovative Skills'], w
                label = 'Creativity and Innovative Skills')
rects6 = ax.bar(x + width, dictionary['Career Opportunity'], width/2, label =

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Priority Of Learning On The Internet vs Academic Institution'
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Priority of learning on the internet', title_fontsize=16, loc

sns.set(font_scale=0.95)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)
autolabel(rects6)

fig.tight_layout()

save_fig('Priority_Of_Learning_On_The_Internet_W.R.T._Academic_Institution_Fre
plt.show()
```

```
Saving figure Priority_Of_Learning_On_The_Internet_W.R.T._Academic_Institution
_Frequency_Distribution
```

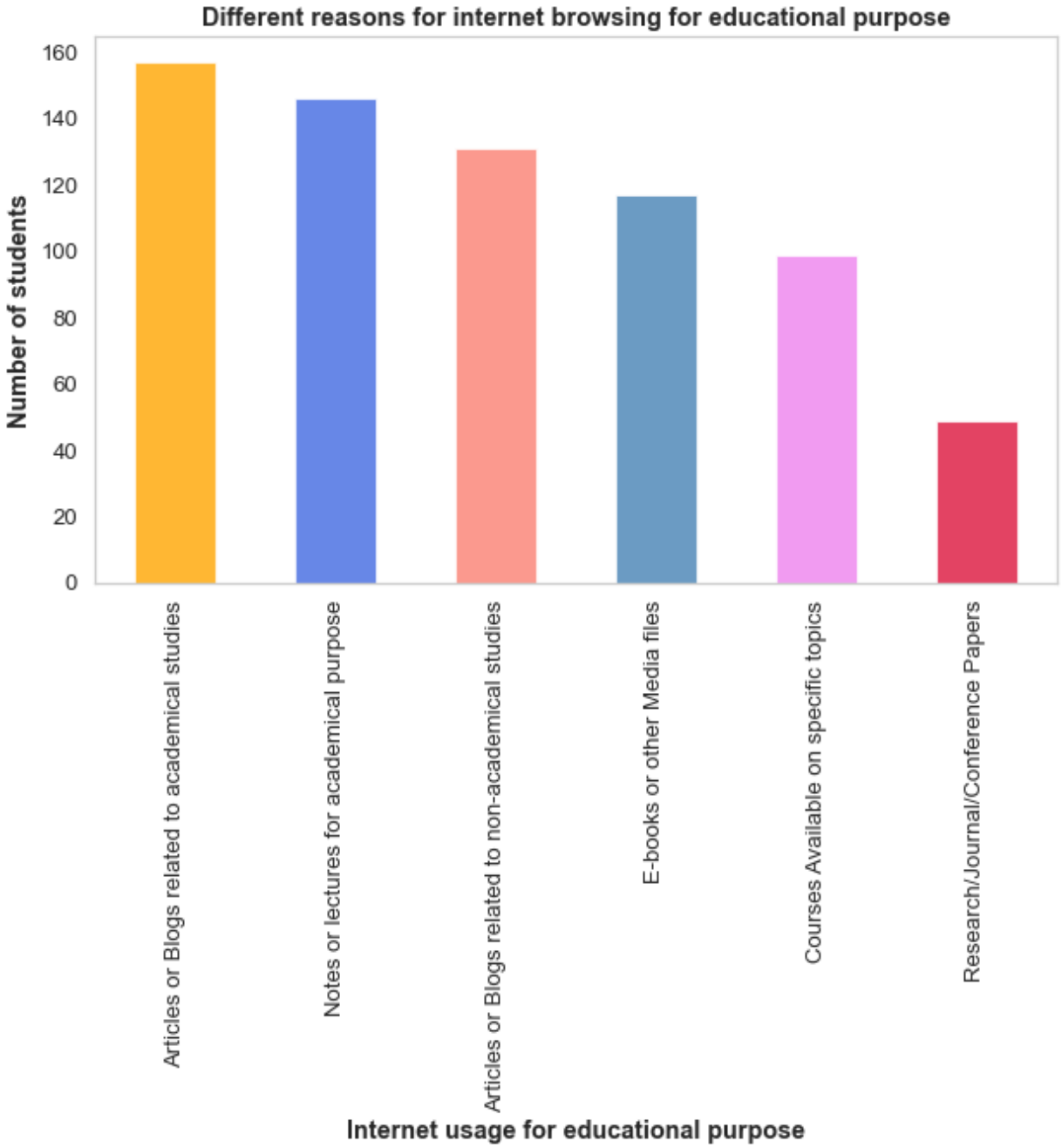


Plotting 'Internet Usage For Educational Purpose'

Let's check the histogram.

In [358…

```
plt.figure(figsize=(10, 11))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Internet Usage For Educational Purpose'], r
                     color=['orange', 'royalblue', 'salmon', 'steelblue', 'vi
                     title='Different reasons for internet browsing for educa
                     xlabel='Internet usage for educational purpose')

plt.show()
```

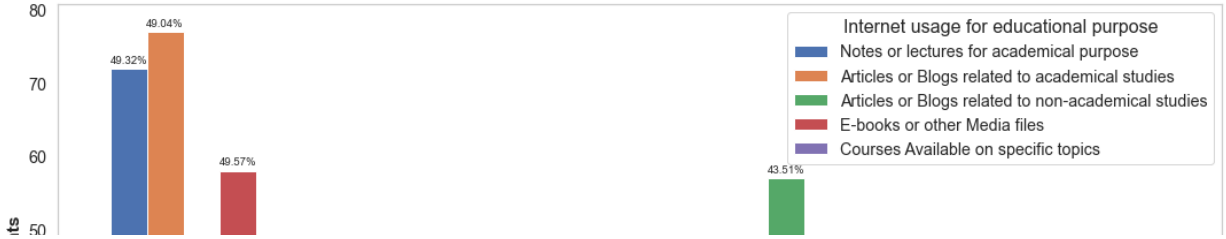


Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [359…

```
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(combined_df, 'Internet Usage For Educational
                                 combined_df['Internet Usage For Educational Purp

labels = ['Excellent', 'Good', 'Average', 'Not Satisfactory']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Notes or lectures for academical purpos
                width/2, label = 'Notes or lectures for academical purpose')
rects2 = ax.bar(x - width/2, dictionary['Articles or Blogs related to academic
                width/2, label = 'Articles or Blogs related to academical stu
rects3 = ax.bar(x, dictionary['Articles or Blogs related to non-academical st
                width/2, label = 'Articles or Blogs related to non-academical
rects4 = ax.bar(x + width/2, dictionary['E-books or other Media files'],
                width/2, label = 'E-books or other Media files')
rects5 = ax.bar(x + width, dictionary['Courses Available on specific topics'],
                width/2, label = 'Courses Available on specific topics')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic performance', fontweight = 'bold')
# ax.set_title('Internet Usage For Educational Purpose vs Academic Performance
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Internet usage for educational purpose', title_fontsize=16, l

sns.set(font_scale=0.8)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)

fig.tight_layout()

save_fig('Internet_Usage_For_Educational_Purpose_WRT_Academic_Performance_Freq
plt.show()
```

```
Saving figure Internet_Usage_For_Educational_Purpose_WRT_Academic_Performance_
Frequency_Distribution
```

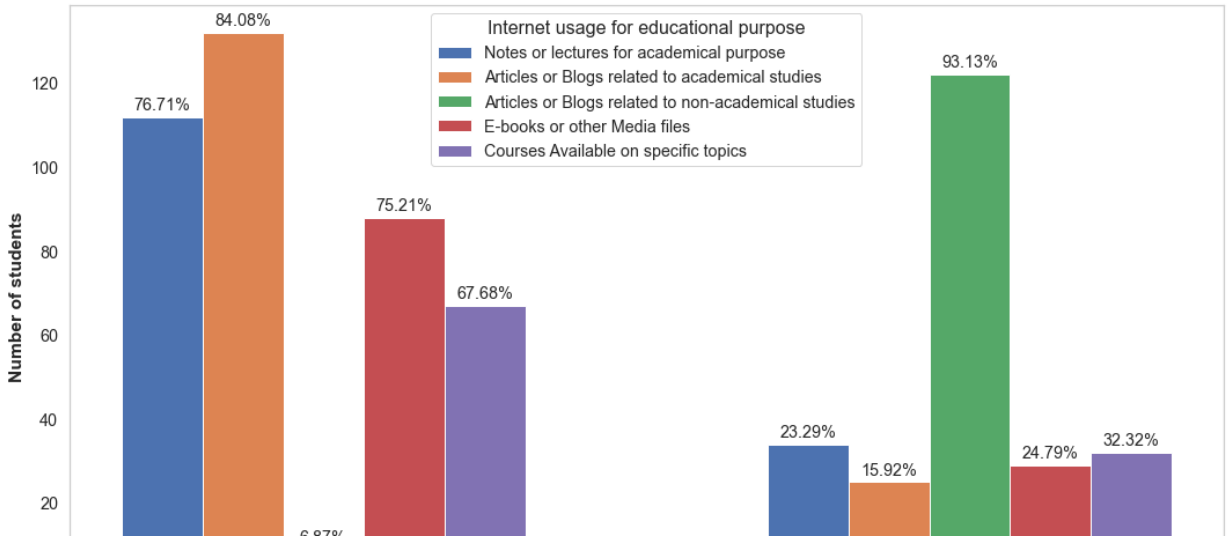


Let's check the distribution of this feature against the target i.e. `'Browsing Purpose'`.

In [360…

```
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_browsing_purpose(combined_df, 'Internet Usage
                                combined_df['Internet Usage For Educational Purp

labels = ['Academic', 'Non-academic']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Notes or lectures for academical purpos
                width/2, label = 'Notes or lectures for academical purpose')
rects2 = ax.bar(x - width/2, dictionary['Articles or Blogs related to academic
                width/2, label = 'Articles or Blogs related to academical stu
rects3 = ax.bar(x, dictionary['Articles or Blogs related to non-academical st
                width/2, label = 'Articles or Blogs related to non-academical
rects4 = ax.bar(x + width/2, dictionary['E-books or other Media files'],
                width/2, label = 'E-books or other Media files')
rects5 = ax.bar(x + width, dictionary['Courses Available on specific topics'],
                width/2, label = 'Courses Available on specific topics')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Browsing purpose', fontweight = 'bold')
# ax.set_title('Internet Usage For Educational Purpose vs Browsing Purpose', f
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Internet usage for educational purpose', title_fontsize=16, l

sns.set(font_scale=1.2)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)

fig.tight_layout()

save_fig('Internet_Usage_For_Educational_Purpose_WRT_Browsing_Purpose_Frequenc
plt.show()
```

```
Saving figure Internet_Usage_For_Educational_Purpose_WRT_Browsing_Purpose_Freq
uency_Distribution
```

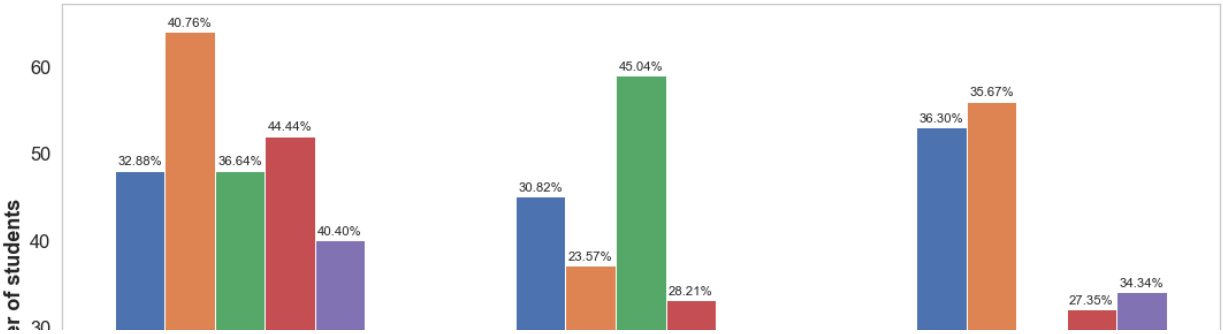


Let's check the distribution of this feature against `'Academic Institution'`.

In [361…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Internet U
                                combined_df['Internet Usage For Educational Pu

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Notes or lectures for academical purpos
                width/2, label = 'Notes or lectures for academical purpose')
rects2 = ax.bar(x - width/2, dictionary['Articles or Blogs related to academic
                width/2, label = 'Articles or Blogs related to academical stud
rects3 = ax.bar(x, dictionary['Articles or Blogs related to non-academical stu
                width/2, label = 'Articles or Blogs related to non-academical
rects4 = ax.bar(x + width/2, dictionary['E-books or other Media files'],
                width/2, label = 'E-books or other Media files')
rects5 = ax.bar(x + width, dictionary['Courses Available on specific topics'],
                width/2, label = 'Courses Available on specific topics')

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Internet Usage For Educational Purpose vs Academic Institutio
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)

sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})
ax.legend(title='Internet usage for educational purpose', title_fontsize=16, l

sns.set(font_scale=0.95)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)
autolabel(rects5)

fig.tight_layout()

save_fig('Internet_Usage_For_Educational_Purpose_WRT_Academic_Institution_Freq
plt.show()
```

```
Saving figure Internet_Usage_For_Educational_Purpose_WRT_Academic_Institution_
Frequency_Distribution
```

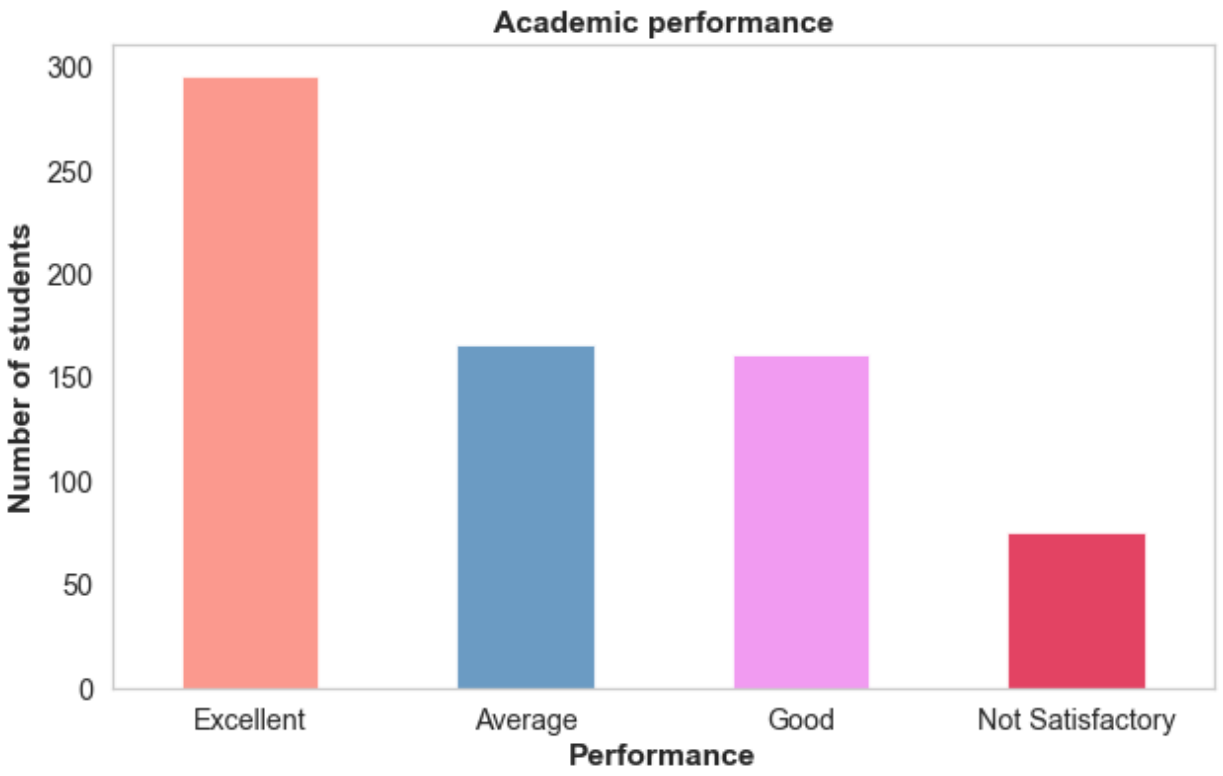


Plotting 'Academic Performance'

Let's check the histogram.

In [362…

```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Academic Performance'], color=['salmon', 'st
                     title='Academic performance', xlabel='Performance')

plt.show()
```



Let's check the distribution of this feature against `'Academic Institution'`.

In [363…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Academic I
                                combined_df['Academic Performance'].value_cour

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['Excellent'], width/2, label = 'Excelle
rects2 = ax.bar(x - width/2, dictionary['Good'], width/2, label = 'Good')
rects3 = ax.bar(x, dictionary['Average'], width/2, label = 'Average')
rects4 = ax.bar(x + width/2, dictionary['Not Satisfactory'], width/2, label =

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Academic Performance vs Academic Institution', fontweight = 'I
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Academic performance', title_fontsize=14)

sns.set(font_scale=0.95)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)
autolabel(rects4)

fig.tight_layout()

save_fig('Academic_Performance_WRT_Academic_Institution_Frequency_Distributio
plt.show()
```

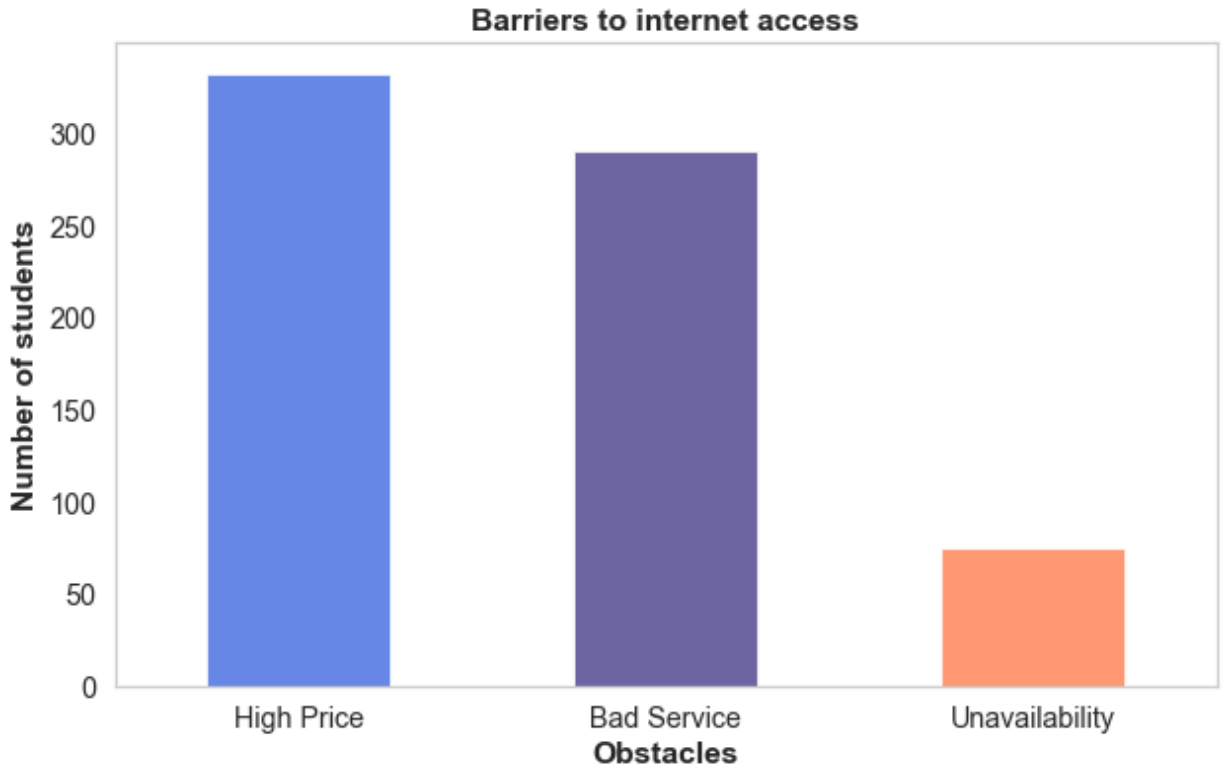
```
Saving figure Academic_Performance_WRT_Academic_Institution_Frequency_Distribu
tion
```

Plotting 'Barriers To Internet Access'

Let's check the histogram.

In [364…

```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(combined_df['Barriers To Internet Access'],
                     color=['royalblue', 'darkslateblue', 'coral', 'crimson'],
                     title='Barriers to internet access', xlabel='Obstacles')

plt.show()
```

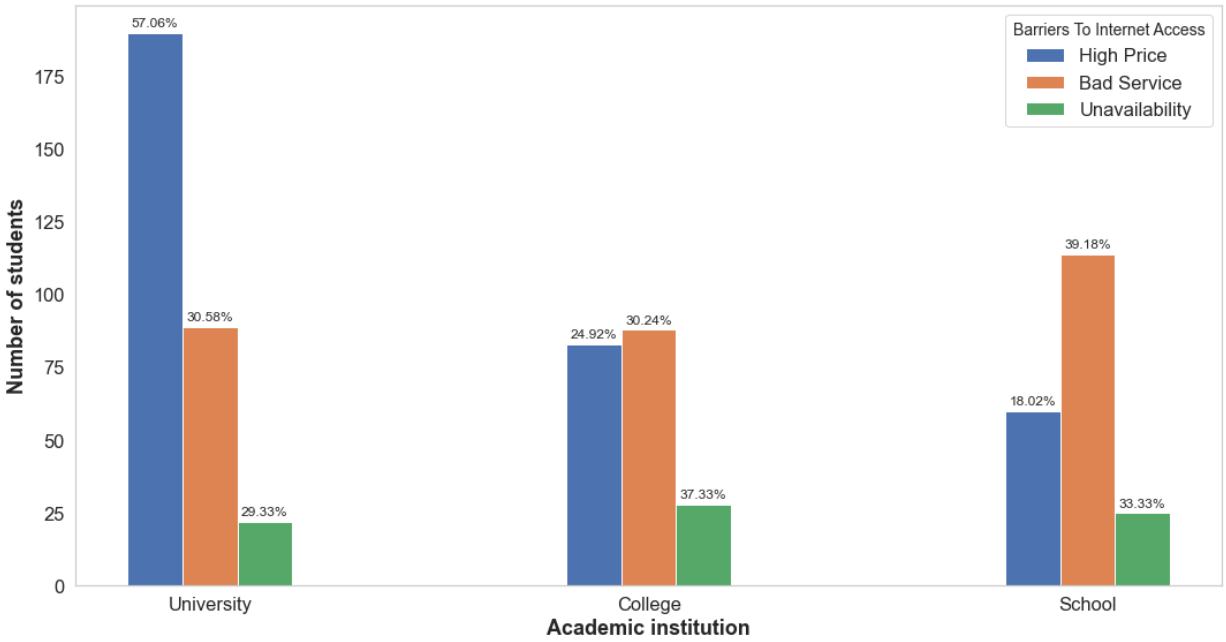


Let's check the distribution of this feature against `'Academic Institution'`.

In [365…

```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_academic_institution(combined_df, 'Barriers T
                                    combined_df['Barriers To Internet Access'].val

labels = ['University', 'College', 'School']
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots(figsize=(15, 8))
fig.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)

rects1 = ax.bar(x - width, dictionary['High Price'], width/2, label = 'High Pr
rects2 = ax.bar(x - width/2, dictionary['Bad Service'], width/2, label = 'Bad
rects3 = ax.bar(x, dictionary['Unavailability'], width/2, label = 'Unavailabi

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Academic institution', fontweight = 'bold')
# ax.set_title('Barriers To Internet Access vs Academic Institution', fontweig
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Barriers To Internet Access', title_fontsize=14)

sns.set(font_scale=1.0)

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

fig.tight_layout()

save_fig('Barriers_To_Internet_Access_WRT_Academic_Institution_Frequency_Distri
plt.show()
```

```
Saving figure Barriers_To_Internet_Access_WRT_Academic_Institution_Frequency_D
istribution
```



Inspecting Age Closer

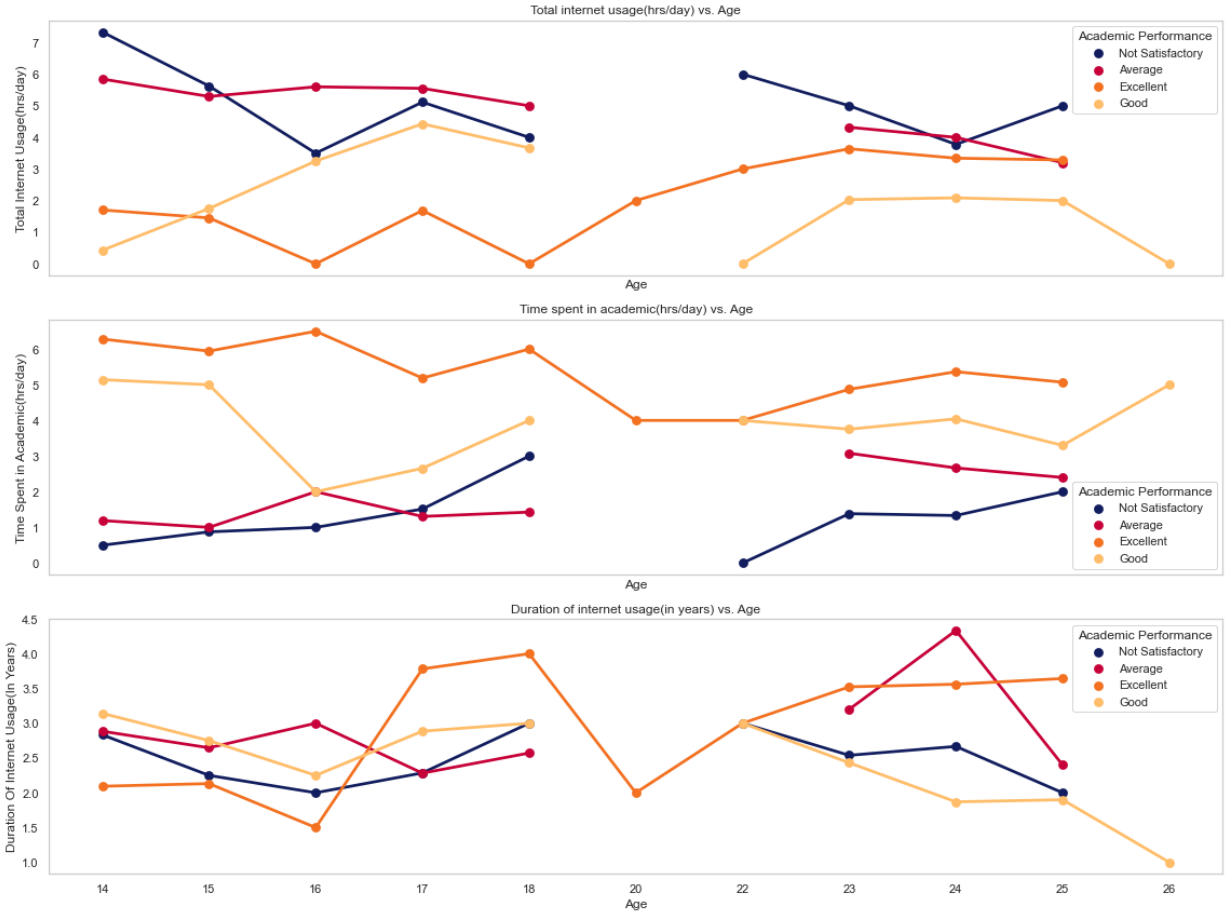
Let's define a function to make this process easier.

In [366…
```
# For Styling:
cust_palt = [
    '#111d5e', '#c70039', '#f37121', '#ffbd69', '#ffc93c'
]
```

In [367…
```
def ctn_freq(dataframe, cols, xax, hue = None, rows = 3, columns = 1):

    sns.set_style("whitegrid", {'axes.grid' : False})

    ''' A function for displaying numerical data frequency vs age and conditi

    fig, axes = plt.subplots(rows, columns, figsize=(16, 12), sharex=True)
    axes = axes.flatten()

    for i, j in zip(dataframe[cols].columns, axes):
        sns.pointplot(x = xax,
                      y = i,
                      data = dataframe,
                      palette = cust_palt[:4],
                      hue = hue,
                      ax = j, ci = False)
        j.set_title(f'{str(i).capitalize()} vs. Age')


        plt.tight_layout()
```

Now let's inspect the columns `'Total Internet Usage(hrs/day)'`, `'Duration Of Internet Usage(In Years)'`, `'Time Spent in Academic(hrs/day)'` against the column `'Age'` and also segment the distribution by the target `'Academic Performance'`.

In [368…
```
ctn_freq(combined_df,
         ['Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
         'Age', hue='Academic Performance',rows=3, columns=1)
```



Multivariate Analysis

Multivariate analysis (MVA) is based on the principles of multivariate statistics, which involves observation and analysis of more than one statistical outcome variable at a time. Typically, MVA is used to address the situations where multiple measurements are made on each experimental unit and the relations among these measurements and their structures are important.

In [369…

```
# Numeric data vs each other and condition:

sns.set(font_scale = 0.7)
sns.set_style("whitegrid", {'axes.grid' : False})

sns.pairplot(combined_df)

plt.show()
```

Let's add `hue = "Academic Performance"` in the `pairplot`

In [370…

```
sns.set(font_scale = 0.7)
sns.set_style("whitegrid", {'axes.grid' : False})

sns.pairplot(combined_df, hue = "Academic Performance")

plt.show()
```

Correlations

We are going to use pearson correlation for to find linear relations between features, heatmap is decent way to show these relations.

In [371…
```
combined_df.corr(method='pearson', min_periods=1)
```

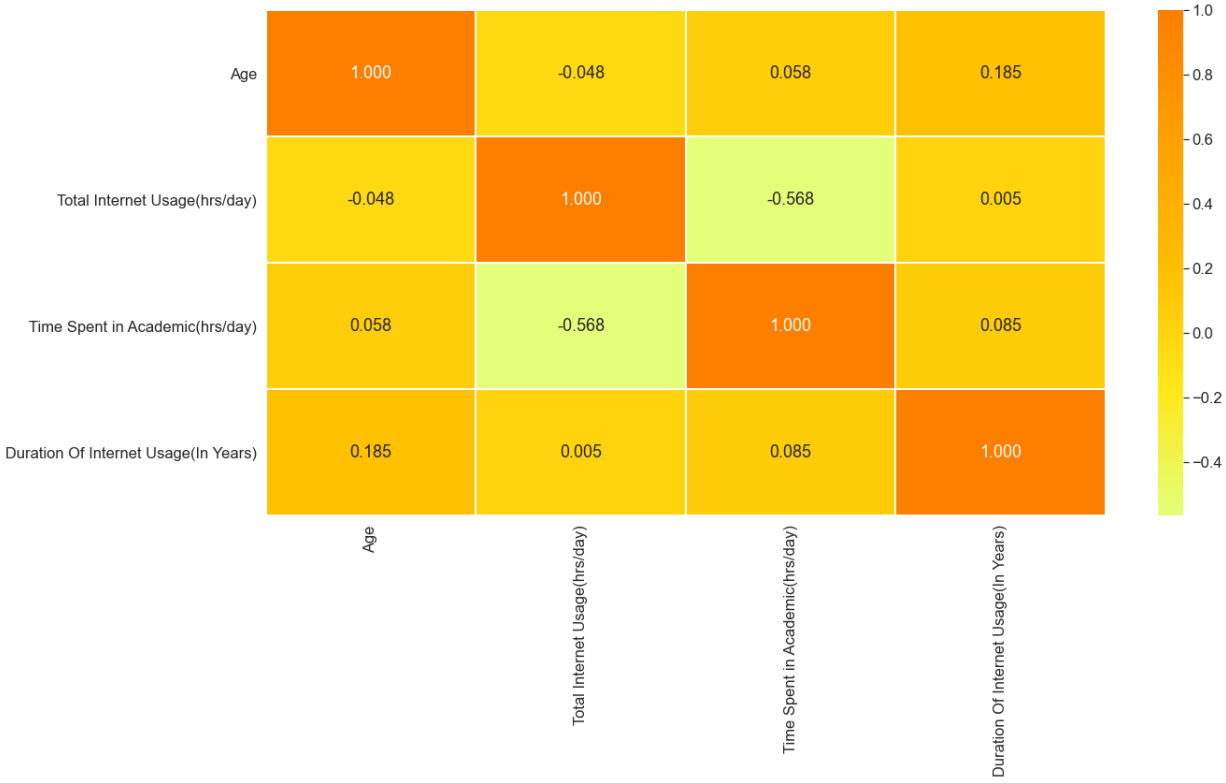
Out[371…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)
Age	1.000000	-0.047796	0.058384	0.185331
Total Internet Usage(hrs/day)	-0.047796	1.000000	-0.567512	0.005265
Time Spent in Academic(hrs/day)	0.058384	-0.567512	1.000000	0.085060
Duration Of Internet Usage(In Years)	0.185331	0.005265	0.085060	1.000000

In [372…

```
# Correlation heatmap between variables:

sns.set(font_scale=1.5)

correlation_df = combined_df.corr(method='pearson', min_periods=1)
mask = np.triu(correlation_df.corr())

plt.figure(figsize=(20, 12))

sns.heatmap(correlation_df,
            annot = True,
            fmt = '.3f',
            cmap = 'Wistia',
            linewidths = 1,
            cbar = True)

save_fig('Correlation_heatmap_of_numerical_variables')
plt.show()
```

```
Saving figure Correlation_heatmap_of_numerical_variables
```



Start Predicting the Models

Let's drop the target column `'Academic Performance'` from the main dataframe. Store the target column on a separate column first.

In [373…

```
labels = combined_df["Academic Performance"].copy()

combined_df.drop("Academic Performance", axis = 1, inplace=True)

combined_df.head()
```

Out[373…

	Gender	Age	Academic Institution	Frequently Visited Website	Effectiveness Of Internet Usage	Devices Used For Internet Browsing	Location Of Internet Use	Household Internet Facilities	Time Of Internet Browsing	F
0	Female	23	University	Instagram	Not at all	Desktop	Library	Connected	Night	
1	Female	23	University	Youtube	Effective	Mobile	University	Connected	Morning	
2	Female	23	University	Whatsapp	Effective	Mobile	University	Connected	Midnight	
3	Female	23	University	Whatsapp	Somewhat Effective	Laptop and Mobile	University	Connected	Morning	
4	Male	24	University	Facebook	Somewhat Effective	Laptop and Mobile	Cyber Cafe	Connected	Night	

In [374…
```
labels.head()
```

Out[374…
```
0    Not Satisfactory
1             Average
2           Excellent
3                Good
4                Good
Name: Academic Performance, dtype: object
```

Let's separate the numerical and categorical columns for preprocessing. Let's check which columns are numerical and which are categorical.

In [375…
```
combined_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 0 to 198
Data columns (total 20 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Gender                                699 non-null    object
 1   Age                                   699 non-null    int64
 2   Academic Institution                  699 non-null    object
 3   Frequently Visited Website            699 non-null    object
 4   Effectiveness Of Internet Usage       699 non-null    object
 5   Devices Used For Internet Browsing    699 non-null    object
 6   Location Of Internet Use              699 non-null    object
 7   Household Internet Facilities         699 non-null    object
 8   Time Of Internet Browsing             699 non-null    object
```

```
 9   Frequency Of Internet Usage             699 non-null    object
 10  Place Of Student's Residence            699 non-null    object
 11  Total Internet Usage(hrs/day)           699 non-null    int64
 12  Time Spent in Academic(hrs/day)         699 non-null    int64
 13  Purpose Of Internet Use                 699 non-null    object
 14  Duration Of Internet Usage(In Years)    699 non-null    int64
 15  Browsing Purpose                        699 non-null    object
 16  Webinar                                 699 non-null    object
 17  Priority Of Learning On The Internet    699 non-null    object
 18  Internet Usage For Educational Purpose  699 non-null    object
 19  Barriers To Internet Access             699 non-null    object
dtypes: int64(4), object(16)
memory usage: 134.7+ KB
```

The columns `'Age'`, `'Total Internet Usage(hrs/day)'`, `'Time Spent in Academic(hrs/day)'`, `'Duration Of Internet Usage(In Years)'` contain numerical values. Let's separate them from the main dataframe.

In [376…

```
combined_cat = combined_df.drop(['Age', 'Total Internet Usage(hrs/day)','Time
                                 'Duration Of Internet Usage(In Years)'], axis = 1)

combined_cat.head()
```

Out[376…

	Gender	Academic Institution	Frequently Visited Website	Effectiveness Of Internet Usage	Devices Used For Internet Browsing	Location Of Internet Use	Household Internet Facilities	Time Of Internet Browsing	Frequer Inter Usa
0	Female	University	Instagram	Not at all	Desktop	Library	Connected	Night	D
1	Female	University	Youtube	Effective	Mobile	University	Connected	Morning	D
2	Female	University	Whatsapp	Effective	Mobile	University	Connected	Midnight	D
3	Female	University	Whatsapp	Somewhat Effective	Laptop and Mobile	University	Connected	Morning	D
4	Male	University	Facebook	Somewhat Effective	Laptop and Mobile	Cyber Cafe	Connected	Night	D

In [377…

```
combined_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 0 to 198
```

```
Data columns (total 16 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Gender                                  699 non-null    object
 1   Academic Institution                    699 non-null    object
 2   Frequently Visited Website              699 non-null    object
 3   Effectiveness Of Internet Usage         699 non-null    object
 4   Devices Used For Internet Browsing      699 non-null    object
 5   Location Of Internet Use                699 non-null    object
 6   Household Internet Facilities           699 non-null    object
 7   Time Of Internet Browsing               699 non-null    object
 8   Frequency Of Internet Usage             699 non-null    object
 9   Place Of Student's Residence            699 non-null    object
 10  Purpose Of Internet Use                 699 non-null    object
 11  Browsing Purpose                        699 non-null    object
 12  Webinar                                 699 non-null    object
 13  Priority Of Learning On The Internet    699 non-null    object
 14  Internet Usage For Educational Purpose  699 non-null    object
 15  Barriers To Internet Access             699 non-null    object
dtypes: object(16)
memory usage: 112.8+ KB
```

Store the numerical attributes in a separate variable.

In [378…
```
combined_num = combined_df[['Age', 'Total Internet Usage(hrs/day)','Time Spent
                            'Duration Of Internet Usage(In Years)']].copy()

combined_num.head()
```

Out[378…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)
0	23	4	2	2
1	23	1	3	2
2	23	5	6	2
3	23	0	4	1
4	24	1	5	3

In [379…
```
combined_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 699 entries, 0 to 198
Data columns (total 4 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   699 non-null    int64
 1   Total Internet Usage(hrs/day)         699 non-null    int64
 2   Time Spent in Academic(hrs/day)       699 non-null    int64
 3   Duration Of Internet Usage(In Years)  699 non-null    int64
dtypes: int64(4)
memory usage: 47.3 KB
```

Let's integerize the categorical values in the dataset `university_cat`. We'll use the `LabelEncoder` from the `sklearn.preprocessing`.

In [380…
```
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

temp_df_cat = combined_cat.apply(preprocessing.LabelEncoder().fit_transform)

temp_df_cat.head()
```

Out[380…

	Gender	Academic Institution	Frequently Visited Website	Effectiveness Of Internet Usage	Devices Used For Internet Browsing	Location Of Internet Use	Household Internet Facilities	Time Of Internet Browsing	Frequen Intern Usa
0	0	2	3	1	0	3	0	3	
1	0	2	6	0	3	6	0	2	
2	0	2	5	0	3	6	0	1	
3	0	2	5	2	2	6	0	2	
4	1	2	0	2	2	1	0	3	

Let's Normalize the dataset using `sklearn`'s `normalize` function. But the dataset seems to perform better without normalization.

In [381…
```
# from sklearn.preprocessing import normalize


# temp_df_normalized = normalize(college_num)
# temp_df_num = pd.DataFrame(temp_df_normalized, columns = list(college_num))

# temp_df_num.head()
```

Let's combine the preprocessed numerical and categorical part of the dataset.

In [382…
```
# Place the DataFrames side by side

X = pd.concat([combined_num, temp_df_cat], axis=1)
y = labels

X.head()
```

Out[382…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)	Gender	Academic Institution	Frequently Visited Website	Effectiveness Of Internet Usage
0	23	4	2	2	0	2	3	1
1	23	1	3	2	0	2	6	0
2	23	5	6	2	0	2	5	0
3	23	0	4	1	0	2	5	2

Duration

Split the dataset for training and testing purposes. We'll use `sklearn`'s `train_test_split` function to do this.

In [383…
```
# split a dataset into train and test sets
from sklearn.model_selection import train_test_split


# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(489, 20) (210, 20) (489,) (210,)
```

Implementing Machine Learning Algorithms For Classification

Stochastic Gradient Descent

Let's start with Stochastic Gradient Descent classifier. We'll use `sklearn`'s `SGDClassifier` to do this. After training the classifier, we'll check the model accuracy score.

In [384…
```
from sklearn.linear_model import SGDClassifier
from sklearn import metrics


sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)

sgd_clf.fit(X_train, y_train)

score = sgd_clf.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.5991820040899796
```

Let's check the confusion matrix and classification report of this model.

In [385…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_sgd = sgd_clf.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_sgd)
class_report = classification_report(y_test, y_pred_sgd)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_sgd))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.5571428571428572
```

```
[[28 12  3  0]
 [ 3 87  4  0]
 [ 8 39  2  0]
 [14  8  2  0]]
                  precision    recall  f1-score   support

         Average       0.53      0.65      0.58        43
       Excellent       0.60      0.93      0.72        94
            Good       0.18      0.04      0.07        49
Not Satisfactory       0.00      0.00      0.00        24

        accuracy                           0.56       210
       macro avg       0.33      0.40      0.34       210
    weighted avg       0.42      0.56      0.46       210

E:\Users\MSI\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:12
21: UndefinedMetricWarning: Precision and F-score are ill-defined and being se
t to 0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Let's perform cross validation using this model. We'll `KFold` for this purpose.

In [386…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_sgd = KFold(n_splits=5, shuffle=True, random_state=42)
cross_val_score(sgd_clf, X_train, y_train, cv=cv_sgd, scoring="accuracy", n_j
```

Out[386…
```
array([0.51020408, 0.62244898, 0.47959184, 0.69387755, 0.51546392])
```

In [387…
```
scores = cross_val_score(sgd_clf, X_test, y_test, cv=cv_sgd, scoring="accuracy
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.514 (0.056)
```

Let's check the score.

In [388…
```
scores = cross_val_score(sgd_clf, X_test, y_test, cv=4, scoring="accuracy", n_
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.514 (0.056)
```

Let's plot the training accuracy curve. But first we'll train and predict the model with `max_iter` in the range of (5, 300)

In [389…

```
m_iter = []
training = []
test = []
scores = {}

max_i = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 70, 80, 90, 100, 130,

for i in range(len(max_i)):
    clf = SGDClassifier(max_iter=max_i[i], tol=1e-3, random_state=42)

    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    m_iter.append(max_i[i])

    training.append(training_score)
    test.append(test_score)
    scores[i] = [training_score, test_score]
```

```
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
```

```
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_stochastic_grad
ient.py:570: ConvergenceWarning: Maximum number of iteration reached before co
nvergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

Let's check the `scores` variable.

In [390…
```
for keys, values in scores.items():
    print(keys, ':', values)
```

```
0 : [0.6605316973415133, 0.6238095238095238]
1 : [0.6155419222903885, 0.5571428571428572]
2 : [0.5378323108384458, 0.5047619047619047]
3 : [0.6175869120654397, 0.5952380952380952]
4 : [0.65439672801636, 0.6142857142857143]
5 : [0.6748466257668712, 0.6238095238095238]
6 : [0.5950920245398773, 0.6047619047619047]
7 : [0.623721881390593, 0.5571428571428572]
8 : [0.6032719836400818, 0.5619047619047619]
9 : [0.6175869120654397, 0.5476190476190477]
10 : [0.6196319018404908, 0.5761904761904761]
11 : [0.6666666666666666, 0.6285714285714286]
12 : [0.5991820040899796, 0.5571428571428572]
13 : [0.5991820040899796, 0.5571428571428572]
14 : [0.5991820040899796, 0.5571428571428572]
15 : [0.5991820040899796, 0.5571428571428572]
16 : [0.5991820040899796, 0.5571428571428572]
17 : [0.5991820040899796, 0.5571428571428572]
18 : [0.5991820040899796, 0.5571428571428572]
19 : [0.5991820040899796, 0.5571428571428572]
20 : [0.5991820040899796, 0.5571428571428572]
21 : [0.5991820040899796, 0.5571428571428572]
```

Finally, let's plot the training score.

In [391…
```
# plt.figure(figsize=(10, 4))
# sns.set(font_scale=1.3)
# sns.set_style("whitegrid", {'axes.grid' : False})

# ax = sns.stripplot(m_iter, training);
# ax.set(xlabel ='max iteration', ylabel ='Training Score')

# plt.show()
```

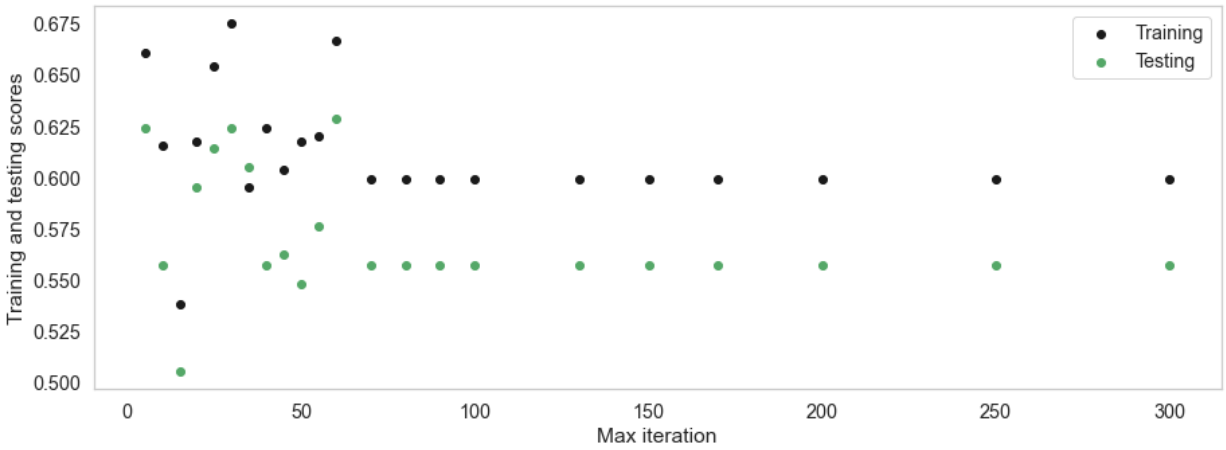
Testing score.

In [392…
```
# plt.figure(figsize=(10, 4))
# sns.set(font_scale=1.3)
# sns.set_style("whitegrid", {'axes.grid' : False})

# ax = sns.stripplot(m_iter, test);
# ax.set(xlabel ='max iteration', ylabel ='Testing Score')

# plt.show()
```

Let's combine the two scores together to compare the two.

In [393…
```
plt.figure(figsize=(13, 5))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.scatter(m_iter, training, color ='k')
plt.scatter(m_iter, test, color ='g')

plt.ylabel('Training and testing scores')
plt.xlabel('Max iteration')
plt.legend(labels=['Training', 'Testing'])

save_fig('SGDClassifier_training_testing_scores')
plt.show()
```

```
Saving figure SGDClassifier_training_testing_scores
```



Decision Tree

Let's start with Decision Tree classifier. We'll use `sklearn`'s `DecisionTreeClassifier` to do this. After training the classifier, we'll check the model accuracy score.

In [394…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

dec_tree_clf = DecisionTreeClassifier(max_depth=20, max_leaf_nodes = 90, rando

dec_tree_clf.fit(X_train, y_train)

score = dec_tree_clf.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.9406952965235174
```

Let's check the confusion matrix and classification report of this model.

In [395…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_dec_tree = dec_tree_clf.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_dec_tree)
class_report = classification_report(y_test, y_pred_dec_tree)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_dec_tree))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.638095238095238
[[27  5  3  8]
 [ 7 80  7  0]
 [11 15 20  3]
 [11  4  2  7]]
                  precision    recall  f1-score   support

         Average       0.48      0.63      0.55        43
       Excellent       0.77      0.85      0.81        94
            Good       0.62      0.41      0.49        49
Not Satisfactory       0.39      0.29      0.33        24

        accuracy                           0.64       210
       macro avg       0.57      0.54      0.55       210
    weighted avg       0.63      0.64      0.63       210
```

Let's perform cross validation using this model. We'll `KFold` for this purpose.

In [396…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_dec_tree = KFold(n_splits=10, shuffle=True, random_state=42)
cross_val_score(dec_tree_clf, X_train, y_train, cv=cv_dec_tree, scoring="accu
```

Out[396…
```
array([0.59183673, 0.69387755, 0.59183673, 0.65306122, 0.59183673,
       0.67346939, 0.69387755, 0.71428571, 0.55102041, 0.6875    ])
```

In [397…
```
scores = cross_val_score(dec_tree_clf, X_test, y_test, cv=cv_dec_tree, scorin
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.510 (0.085)
```

Let's check the score.

In [398…
```
scores = cross_val_score(dec_tree_clf, X_test, y_test, cv=3, scoring="accuracy
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.543 (0.012)
```

Let's plot the training accuracy curve. But first we'll train and predict the model with `max_depth` in the range of (1, 27)

In [399…
```
m_depth = []
training = []
test = []
scores = {}

max_d = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 2(

for i in range(len(max_d)):
    clf = DecisionTreeClassifier(max_depth=max_d[i], max_leaf_nodes = 50, ranc

    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    m_depth.append(max_d[i])

    training.append(training_score)
    test.append(test_score)
    scores[i] = [training_score, test_score]
```

Let's check the `scores` variable.

In [400…
```
for keys, values in scores.items():
    print(keys, ':', values)
```

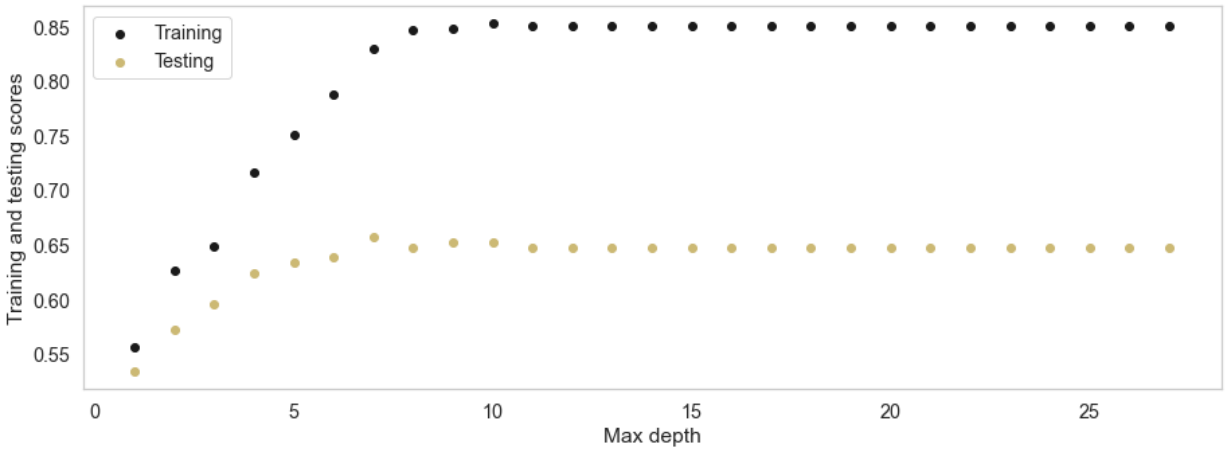
```
0 : [0.556237218813906, 0.5333333333333333]
1 : [0.6257668711656442, 0.5714285714285714]
2 : [0.6482617586912065, 0.5952380952380952]
3 : [0.7157464212678937, 0.6238095238095238]
4 : [0.7505112474437627, 0.6333333333333333]
5 : [0.787321063394683, 0.638095238095238]
6 : [0.8302658486707567, 0.6571428571428571]
7 : [0.8466257668711656, 0.6476190476190476]
8 : [0.8486707566462167, 0.6523809523809524]
9 : [0.852760736196319, 0.6523809523809524]
10 : [0.8507157464212679, 0.6476190476190476]
11 : [0.8507157464212679, 0.6476190476190476]
12 : [0.8507157464212679, 0.6476190476190476]
13 : [0.8507157464212679, 0.6476190476190476]
14 : [0.8507157464212679, 0.6476190476190476]
15 : [0.8507157464212679, 0.6476190476190476]
16 : [0.8507157464212679, 0.6476190476190476]
17 : [0.8507157464212679, 0.6476190476190476]
18 : [0.8507157464212679, 0.6476190476190476]
19 : [0.8507157464212679, 0.6476190476190476]
20 : [0.8507157464212679, 0.6476190476190476]
21 : [0.8507157464212679, 0.6476190476190476]
22 : [0.8507157464212679, 0.6476190476190476]
23 : [0.8507157464212679, 0.6476190476190476]
24 : [0.8507157464212679, 0.6476190476190476]
25 : [0.8507157464212679, 0.6476190476190476]
26 : [0.8507157464212679, 0.6476190476190476]
```

Finally, let's plot the training and testing scores together so that we can compare the two.

In [401…

```
plt.figure(figsize=(13, 5))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.scatter(m_depth, training, color ='k')
plt.scatter(m_depth, test, color ='y')

plt.ylabel('Training and testing scores')
plt.xlabel('Max depth')
plt.legend(labels=['Training', 'Testing'])

save_fig('DecisionTreeClassifier_training_testing_scores')
plt.show()
```

```
Saving figure DecisionTreeClassifier_training_testing_scores
```



Logistic Regression

Let's start with Logistic Regression classifier. We'll use `sklearn`'s `LogisticRegression` to do this. After training the classifier, we'll check the model accuracy score.

In [402…

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

log_reg = LogisticRegression(max_iter=1000, multi_class='multinomial', random_

log_reg.fit(X_train, y_train)

score = log_reg.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.7096114519427403
```

Let's check the confusion matrix and classification report of this model.

In [403…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_log = log_reg.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_log)
class_report = classification_report(y_test, y_pred_log)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_log))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.638095238095238
[[27  5  9  2]
 [ 7 80  7  0]
 [ 6 16 25  2]
 [14  3  5  2]]
                  precision    recall  f1-score   support

         Average       0.50      0.63      0.56        43
       Excellent       0.77      0.85      0.81        94
            Good       0.54      0.51      0.53        49
Not Satisfactory       0.33      0.08      0.13        24

        accuracy                           0.64       210
       macro avg       0.54      0.52      0.51       210
    weighted avg       0.61      0.64      0.61       210
```

Let's perform cross validation using this model. We'll `KFold` for this purpose.

In [404…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_log_reg = KFold(n_splits=10, shuffle=True, random_state=42)
cross_val_score(log_reg, X_train, y_train, cv=cv_log_reg, scoring="accuracy",
```

Out[404…
```
array([0.71428571, 0.75510204, 0.67346939, 0.71428571, 0.46938776,
       0.63265306, 0.71428571, 0.75510204, 0.57142857, 0.6875    ])
```

In [405…
```
scores = cross_val_score(log_reg, X_test, y_test, cv=cv_log_reg, scoring="accu
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.576 (0.086)
```

Let's check the score.

In [406…
```
scores = cross_val_score(log_reg, X_test, y_test, cv=5, scoring="accuracy", n_
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.581 (0.056)
```

Let's plot the training accuracy curve. But first we'll train and predict the model with `max_iter` in the range of (50, 200)

In [407…

```
m_iter = []
training = []
test = []
scores = {}

max_i = [50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
         1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000]
#          22, 23, 24, 25, 26, 27]

for i in range(len(max_i)):
    clf = LogisticRegression(max_iter=max_i[i], multi_class='multinomial', ran

    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    m_iter.append(max_i[i])

    training.append(training_score)
    test.append(test_score)
    scores[i] = [training_score, test_score]
```

```
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
```

```
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
  n_iter_i = _check_optimize_result(
E:\Users\MSI\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
2: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regress
ion
```

Let's check the `scores` variable.

In [408…

```
for keys, values in scores.items():
    print(keys, ':', values)
```

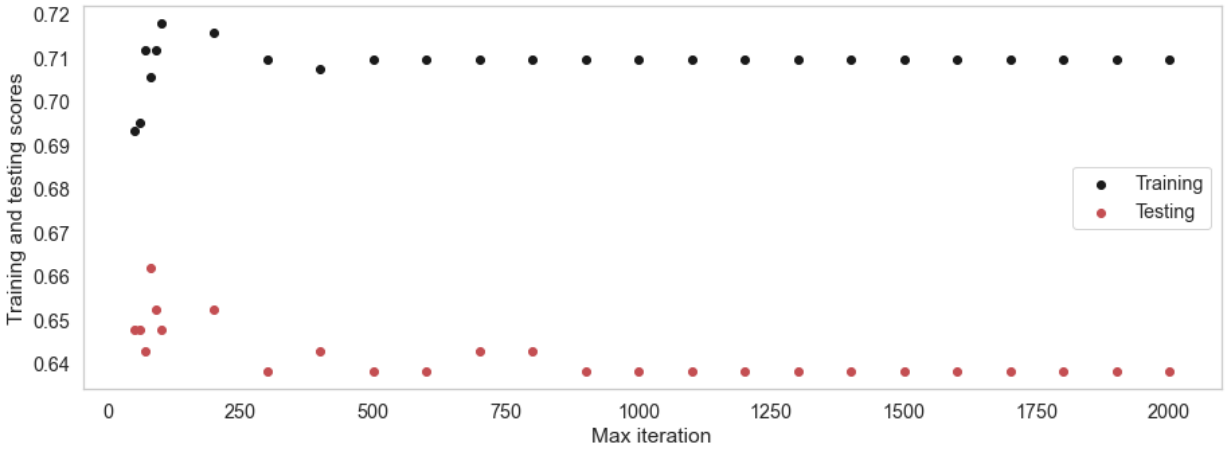
```
0 : [0.6932515337423313, 0.6476190476190476]
1 : [0.6952965235173824, 0.6476190476190476]
2 : [0.7116564417177914, 0.6428571428571429]
3 : [0.7055214723926381, 0.6619047619047619]
4 : [0.7116564417177914, 0.6523809523809524]
5 : [0.7177914110429447, 0.6476190476190476]
6 : [0.7157464212678937, 0.6523809523809524]
7 : [0.7096114519427403, 0.638095238095238]
8 : [0.7075664621676891, 0.6428571428571429]
9 : [0.7096114519427403, 0.638095238095238]
10 : [0.7096114519427403, 0.638095238095238]
11 : [0.7096114519427403, 0.6428571428571429]
12 : [0.7096114519427403, 0.6428571428571429]
13 : [0.7096114519427403, 0.638095238095238]
14 : [0.7096114519427403, 0.638095238095238]
15 : [0.7096114519427403, 0.638095238095238]
16 : [0.7096114519427403, 0.638095238095238]
17 : [0.7096114519427403, 0.638095238095238]
18 : [0.7096114519427403, 0.638095238095238]
19 : [0.7096114519427403, 0.638095238095238]
20 : [0.7096114519427403, 0.638095238095238]
21 : [0.7096114519427403, 0.638095238095238]
22 : [0.7096114519427403, 0.638095238095238]
23 : [0.7096114519427403, 0.638095238095238]
24 : [0.7096114519427403, 0.638095238095238]
```

Finally, let's plot the training and testing scores together so that we can compare the two.

In [409…

```
plt.figure(figsize=(13, 5))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.scatter(m_iter, training, color ='k')
plt.scatter(m_iter, test, color ='r')

plt.ylabel('Training and testing scores')
plt.xlabel('Max iteration')
plt.legend(labels=['Training', 'Testing'])

save_fig('LogisticRegression_training_testing_scores')
plt.show()
```

```
Saving figure LogisticRegression_training_testing_scores
```



Random Forest

Let's start with Random Forest classifier. We'll use `sklearn`'s `RandomForestClassifier` to do this. After training the classifier, we'll check the model accuracy score.

In [410…

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

random_for_clf = RandomForestClassifier(n_estimators=13, max_depth=100, randor

random_for_clf.fit(X_train, y_train)

score = random_for_clf.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.9897750511247444
```

Let's check the confusion matrix and classification report of this model.

In [411…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_rand = random_for_clf.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_rand)
class_report = classification_report(y_test, y_pred_rand)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_rand))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.6857142857142857
[[31  7  3  2]
 [ 3 86  5  0]
 [ 9 18 21  1]
 [11  3  4  6]]
                  precision    recall  f1-score   support

         Average       0.57      0.72      0.64        43
       Excellent       0.75      0.91      0.83        94
            Good       0.64      0.43      0.51        49
Not Satisfactory       0.67      0.25      0.36        24

        accuracy                           0.69       210
       macro avg       0.66      0.58      0.59       210
    weighted avg       0.68      0.69      0.66       210
```

Let's perform cross validation using this model. We'll `KFold` for this purpose.

In [412…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_rand_for = KFold(n_splits=5, shuffle=True, random_state=42)
cross_val_score(random_for_clf, X_train, y_train, cv=cv_rand_for, scoring="ac
```

Out[412…
```
array([0.69387755, 0.68367347, 0.65306122, 0.70408163, 0.71134021])
```

In [413…
```
scores = cross_val_score(random_for_clf, X_test, y_test, cv=cv_rand_for, scor
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.576 (0.065)
```

Let's check the score.

In [414…
```
scores = cross_val_score(random_for_clf, X_test, y_test, cv=4, scoring="accura
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.629 (0.015)
```

Let's plot the training accuracy curve. But first we'll train and predict the model with `n_estimators` in the range of (1, 35)

In [415…
```
n_estimate = []
training = []
test = []
scores = {}

for i in range(1, 35):
    clf = RandomForestClassifier(n_estimators=i, max_depth=50, random_state=4

    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    n_estimate.append(i)

    training.append(training_score)
    test.append(test_score)
    scores[i] = [training_score, test_score]
```

Let's check the `scores` variable.

In [416…
```
for keys, values in scores.items():
    print(keys, ':', values)
```

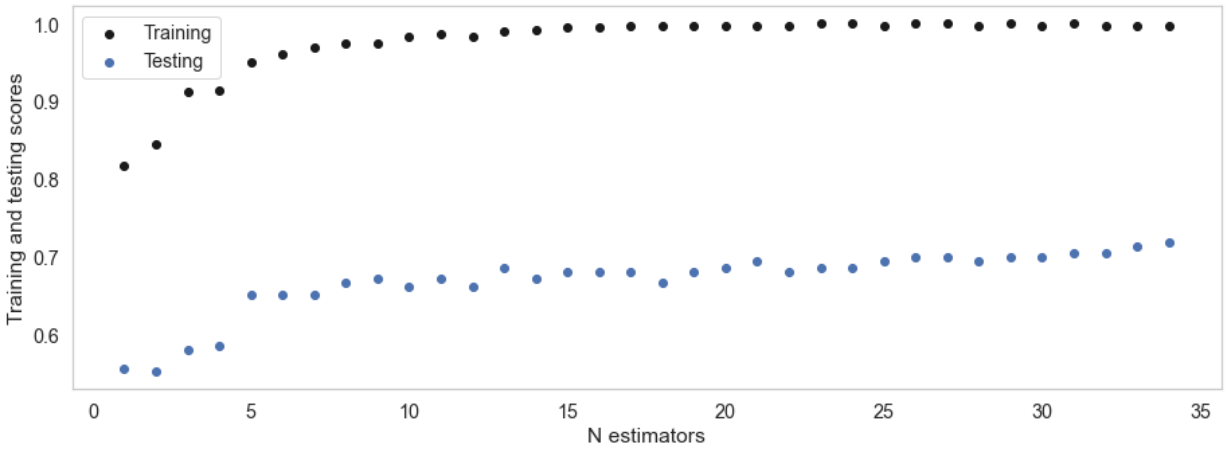
```
1 : [0.8179959100204499, 0.5571428571428572]
2 : [0.8445807770961146, 0.5523809523809524]
3 : [0.9120654396728016, 0.580952380952381]
4 : [0.9141104294478528, 0.5857142857142857]
5 : [0.950920245398773, 0.6523809523809524]
6 : [0.9611451942740287, 0.6523809523809524]
7 : [0.9693251533742331, 0.6523809523809524]
8 : [0.9754601226993865, 0.6666666666666666]
9 : [0.9754601226993865, 0.6714285714285714]
10 : [0.983640081799591, 0.6619047619047619]
11 : [0.9877300613496932, 0.6714285714285714]
12 : [0.983640081799591, 0.6619047619047619]
13 : [0.9897750511247444, 0.6857142857142857]
14 : [0.9918200408997955, 0.6714285714285714]
15 : [0.9959100204498977, 0.680952380952381]
16 : [0.9959100204498977, 0.680952380952381]
17 : [0.9979550102249489, 0.680952380952381]
18 : [0.9979550102249489, 0.6666666666666666]
19 : [0.9979550102249489, 0.680952380952381]
20 : [0.9979550102249489, 0.6857142857142857]
21 : [0.9979550102249489, 0.6952380952380952]
22 : [0.9979550102249489, 0.680952380952381]
23 : [1.0, 0.6857142857142857]
24 : [1.0, 0.6857142857142857]
25 : [0.9979550102249489, 0.6952380952380952]
26 : [1.0, 0.7]
27 : [1.0, 0.7]
28 : [0.9979550102249489, 0.6952380952380952]
29 : [1.0, 0.7]
30 : [0.9979550102249489, 0.7]
31 : [1.0, 0.7047619047619048]
32 : [0.9979550102249489, 0.7047619047619048]
33 : [0.9979550102249489, 0.7142857142857143]
34 : [0.9979550102249489, 0.719047619047619]
```

Finally, let's plot the training and testing scores together so that we can compare the two.

In [417…

```
plt.figure(figsize=(13, 5))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.scatter(n_estimate, training, color ='k')
plt.scatter(n_estimate, test, color ='b')

plt.ylabel('Training and testing scores')
plt.xlabel('N estimators')
plt.legend(labels=['Training', 'Testing'])

save_fig('RandomForestClassifier_training_testing_scores')
plt.show()
```

```
Saving figure RandomForestClassifier_training_testing_scores
```



Naive Bayes

Let's start with Naive Bayes classifier. We'll use `sklearn`'s `GaussianNB`, `MultinomialNB` and `CategoricalNB` to do this. After training the classifier, we'll check the model accuracy score.

In [418…

```
### 1.GaussianNB
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics


gaussNB_clf = GaussianNB()

gaussNB_clf.fit(X_train, y_train)

score = gaussNB_clf.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.6462167689161554
```

In [419…
```
### 2.MultinomialNB
from sklearn.naive_bayes import MultinomialNB


multinomNB_clf = MultinomialNB()

multinomNB_clf.fit(X_train, y_train)

score = multinomNB_clf.score(X_train, y_train)
print("Training score: ", score)
```

```
Training score:  0.6421267893660532
```

Both `GaussianNB` and `MultinomialNB` have the same training accuracy.

Let's check the confusion matrix and classification report of `GaussianNB` model.

In [420…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_nb = gaussNB_clf.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_nb)
class_report = classification_report(y_test, y_pred_nb)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_nb))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.5761904761904761
[[25  3 11  4]
 [ 9 66 18  1]
 [ 9 12 28  0]
 [14  1  7  2]]
                  precision    recall  f1-score   support

         Average       0.44      0.58      0.50        43
       Excellent       0.80      0.70      0.75        94
            Good       0.44      0.57      0.50        49
Not Satisfactory       0.29      0.08      0.13        24

        accuracy                           0.58       210
       macro avg       0.49      0.48      0.47       210
    weighted avg       0.58      0.58      0.57       210
```

Let's perform cross validation using this model. We'll `KFold` for this purpose.

In [421…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_gauss_nb = KFold(n_splits=15, shuffle=True, random_state=42)
cross_val_score(gaussNB_clf, X_train, y_train, cv=cv_gauss_nb, scoring="accura
```

Out[421…
```
array([0.75757576, 0.57575758, 0.63636364, 0.66666667, 0.63636364,
       0.66666667, 0.45454545, 0.45454545, 0.72727273, 0.625     ,
       0.6875    , 0.65625   , 0.5       , 0.625     , 0.65625   ])
```

In [422…
```
scores = cross_val_score(gaussNB_clf, X_test, y_test, cv=cv_gauss_nb, scoring=
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.524 (0.144)
```

Let's check the confusion matrix and classification report of this model.

In [423…
```
scores = cross_val_score(gaussNB_clf, X_test, y_test, cv=5, scoring="accuracy"
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.500 (0.043)
```

Check Feature Importance

Univariate Selection

Statistical tests can be used to select those features that have the strongest relationship with the output variable. The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features. The code below uses the chi-squared (chi^2) statistical test for non-negative features to select 10 of the best features from the Mobile Price Range Prediction Dataset.

In [201…
```
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2


bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X, y)

dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)

#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns, dfscores], axis=1)
featureScores.columns = ['Specs', 'Score']  #naming the dataframe columns

print(featureScores.nlargest(10, 'Score'))  #print 10 best features
```

```
                                  Specs       Score
2        Time Spent in Academic(hrs/day)  570.040553
1          Total Internet Usage(hrs/day)  337.137883
17  Priority Of Learning On The Internet  142.415804
14               Purpose Of Internet Use   88.534002
15                       Browsing Purpose   78.366189
5                   Academic Institution   64.994511
7        Effectiveness Of Internet Usage   45.682749
0                                    Age   42.650318
16                               Webinar   33.401205
10         Household Internet Facilities   29.354177
```

Feature Importance

We can get the feature importance of each feature of our dataset by using the feature importance property of the model. Feature importance gives a score for each feature of the data, the higher the score more important or relevant is the feature towards our output variable. Feature importance is an inbuilt class that comes with Tree Based Classifiers, we will be using

Extra Tree Classifier for extracting the top 10 features for the dataset.

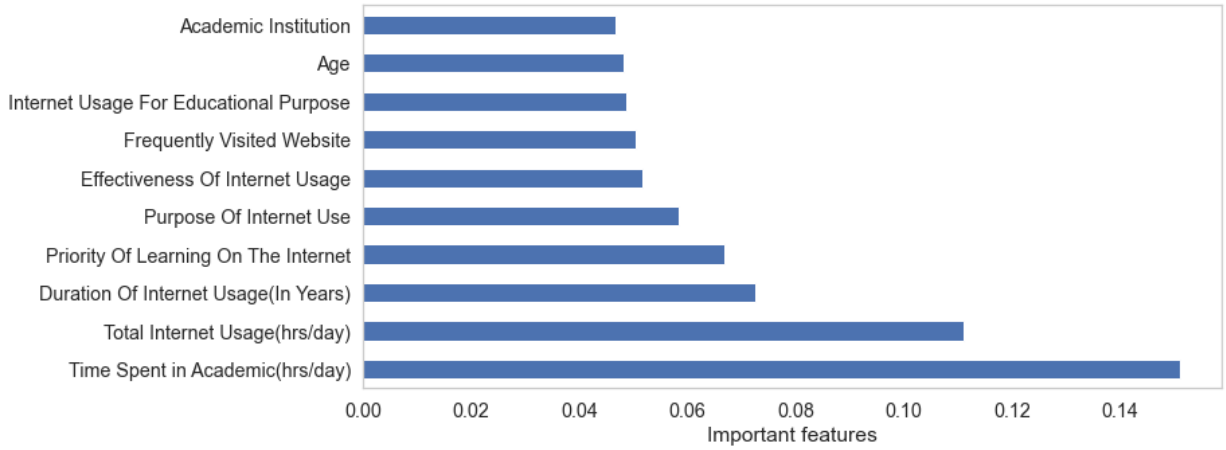
In [202…
```
import pandas as pd
import numpy as np
from sklearn.ensemble import ExtraTreesClassifier
import matplotlib.pyplot as plt


model = ExtraTreesClassifier()
model.fit(X, y)
print(model.feature_importances_) #use inbuilt class feature_importances of t

#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index = X.columns)
```

```
[0.04820506 0.11099792 0.15112311 0.07256845 0.02615489 0.0466751
 0.05042633 0.05156178 0.02853165 0.02727831 0.02497425 0.03128621
 0.0250181  0.02733654 0.05843754 0.04422931 0.02187332 0.06679648
 0.04858018 0.03794546]
```

Let's plot the top 10 most important features.

In [203…
```
plt.figure(figsize=(13, 5))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

feat_importances.nlargest(10).plot(kind='barh')

plt.xlabel('Important features')

save_fig('top_ten_important_features')
plt.show()
```

```
Saving figure top_ten_important_features
```

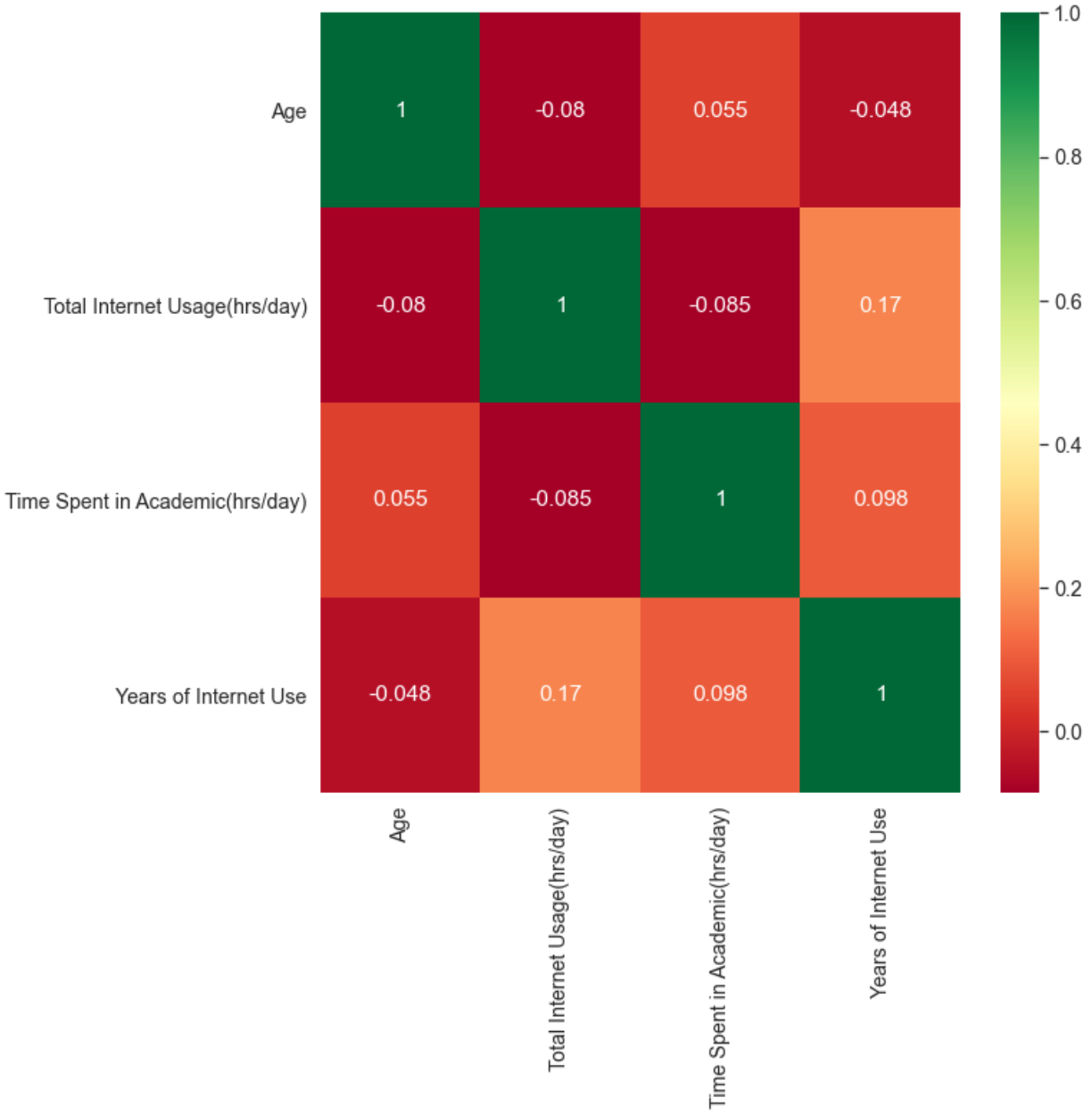


Correlation Matrix with Heatmap

Correlation states how the features are related to each other or the target variable. Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable) Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

In [204…

```
import pandas as pd
import numpy as np
import seaborn as sns

#get correlations of each features in dataset
corrmat = university_df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(10,10))

#plot heat map
g=sns.heatmap(university_df[top_corr_features].corr(),annot=True,cmap="RdYlGn"
```



Hyperparameter Optimization

hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node

weights) are learned.
We'll perform hyperparameter optimization using the following optimization techniques:

1. **GridSearchCV** - Exhaustive search over specified parameter values for an estimator.

1. **RandomizedSearchCV** - Randomized search on hyper parameters. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

1. **BayesSearchCV** - Bayesian Optimization of model hyperparameters provided by the Scikit-Optimize library.

1. **Genetic Algorithm using the TPOT library** - TPOT is an open-source library for performing AutoML in Python. It makes use of the popular Scikit-Learn machine learning library for data transforms and machine learning algorithms and uses a Genetic Programming stochastic global search procedure to efficiently discover a top-performing model pipeline for a given dataset.

Let's start with `GridSearchCV`.

Hyperparameter Optimization using `GridSearchCV`

As we saw, the algorithms that performs the best is the `LogisticRegression` and `RandomForestClassifier`. Let's try and optimize the `RandomForestClassifier` algorithm more to get a better result. First let's see the parameters that we'll try and tune in the `RandomForestClassifier`.

In [205…
```
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics


random_for_clf = RandomForestClassifier()

random_for_clf.get_params().keys()
```

Out[205…
```
dict_keys(['bootstrap', 'ccp_alpha', 'class_weight', 'criterion', 'max_depth',
'max_features', 'max_leaf_nodes', 'max_samples', 'min_impurity_decrease', 'min
_impurity_split', 'min_samples_leaf', 'min_samples_split', 'min_weight_fractio
n_leaf', 'n_estimators', 'n_jobs', 'oob_score', 'random_state', 'verbose', 'wa
rm_start'])
```

Let's create a dictionary that defines the parameters that we want to optimize.

In [206…
```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 50, stop = 250, num = 5)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 50, num = 10)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]# Create the random grid


random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap
              }

print(random_grid)
```

```
{'n_estimators': [50, 100, 150, 200, 250], 'max_features': ['auto', 'sqrt'], '
max_depth': [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, None], 'min_samples_split
': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'bootstrap': [True, False]}
```

Now, let's optimize the model using `GridSearchCV` . The method we'll use for cross validation is `RepeatedStratifiedKFold` .

In [207…
```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold

# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

# define the search
gs_rand_for = GridSearchCV(random_for_clf, param_grid=random_grid, scoring='a

gs_rand_for.fit(X_train, y_train)

gs_rand_for.best_params_
```

Out[207…
```
{'bootstrap': False,
 'max_depth': 10,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 250}
```

Let's check the training score. It should be performing much better now.

In [208…
```
gs_rand_for.score(X_train, y_train)
```

Out[208…
```
1.0
```

Let's put the model to use and predict our test set.

In [209…
```
y_pred_gs_rand = gs_rand_for.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_gs_rand)
class_report = classification_report(y_test, y_pred_gs_rand)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_gs_rand))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.7238095238095238
[[37  6  4  4]
 [ 2 78  4  1]
 [ 7 12 26  1]
 [14  3  0 11]]
                  precision    recall  f1-score   support

         Average       0.62      0.73      0.67        51
       Excellent       0.79      0.92      0.85        85
            Good       0.76      0.57      0.65        46
Not Satisfactory       0.65      0.39      0.49        28

        accuracy                           0.72       210
       macro avg       0.70      0.65      0.66       210
    weighted avg       0.72      0.72      0.71       210
```

Hyperparameter Optimization using RandomizedSearchCV

As we saw, the algorithms that performs the best is the `LogisticRegression` and `RandomForestClassifier`. Let's try and optimize the `RandomForestClassifier` algorithm more to get a better result. First let's see the parameters that we'll try and tune in the `RandomForestClassifier`.

We'll use the same dictionary that we created before as the parameters that we want to optimize. Now, let's optimize the model using `RandomizedSearchCV`. The method we'll use for cross validation is `RepeatedStratifiedKFold`.

In [210…
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform

# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

rs_rand_for = RandomizedSearchCV(random_for_clf, random_grid, scoring='accura

rs_rand_for.fit(X_train, y_train)

rs_rand_for.best_params_
```

Out[210…
```
{'n_estimators': 200,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
```

```
 'max_features': 'auto',
 'max_depth': 45,
```

Let's check the training score. It should be performing much better now.

In [211…
```
rs_rand_for.score(X_train, y_train)
```

Out[211… 1.0

Let's put the model to use and predict our test set.

In [212…
```
y_pred_rs_rand = rs_rand_for.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_rs_rand)
class_report = classification_report(y_test, y_pred_rs_rand)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_rs_rand))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.7285714285714285
[[37  4  4  6]
 [ 3 79  2  1]
 [ 7 11 26  2]
 [12  4  1 11]]
                  precision    recall  f1-score   support

         Average       0.63      0.73      0.67        51
       Excellent       0.81      0.93      0.86        85
            Good       0.79      0.57      0.66        46
Not Satisfactory       0.55      0.39      0.46        28

        accuracy                           0.73       210
       macro avg       0.69      0.65      0.66       210
    weighted avg       0.72      0.73      0.72       210
```

Hyperparameter Optimization using BayesSearchCV

As we saw, the algorithms that performs the best is the `LogisticRegression` and `RandomForestClassifier`. Let's try and optimize the `RandomForestClassifier` algorithm more to get a better result. First let's see the parameters that we'll try and tune in the `RandomForestClassifier`.

we'll use the same dictionary that we created before as the parameters that we want to optimize. Now, let's optimize the model using **Bayesian Optimization** implemented in `BayesSearchCV`. `skopt` library contains this class. The method we'll use for cross validation is `RepeatedStratifiedKFold`.

In [213…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from skopt import BayesSearchCV


# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

# define the search
bs_rand_for = BayesSearchCV(estimator=random_for_clf, search_spaces=random_gr

# perform the search
bs_rand_for.fit(X, y)

# report the best result
print(bs_rand_for.best_score_)
print(bs_rand_for.best_params_)
```

```
0.7406832298136646
OrderedDict([('bootstrap', False), ('max_depth', 45), ('max_features', 'auto
'), ('min_samples_leaf', 1), ('min_samples_split', 2), ('n_estimators', 250)])
```

Let's check the training score. It should be performing much better now.

In [214…
```
bs_rand_for.score(X_train, y_train)
```

Out[214… 1.0

Let's put the model to use and predict our test set.

In [215…
```
y_pred_bs_rand = bs_rand_for.predict(X_test)

conf_mat = confusion_matrix(y_test, y_pred_bs_rand)
class_report = classification_report(y_test, y_pred_bs_rand)


print("Accuracy:", metrics.accuracy_score(y_test, y_pred_bs_rand))

print(conf_mat)
print(class_report)
```

```
Accuracy: 1.0
[[51  0  0  0]
 [ 0 85  0  0]
 [ 0  0 46  0]
 [ 0  0  0 28]]
                  precision    recall  f1-score   support

         Average       1.00      1.00      1.00        51
       Excellent       1.00      1.00      1.00        85
            Good       1.00      1.00      1.00        46
Not Satisfactory       1.00      1.00      1.00        28

        accuracy                           1.00       210
       macro avg       1.00      1.00      1.00       210
    weighted avg       1.00      1.00      1.00       210
```

Hyperparameter Optimization using `Genetic`

Algorithm

Genetic Algorithms(GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

Genetic algorithms simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate “survival of the fittest” among individual of consecutive generation for solving a problem. Each generation consist of a population of individuals and each individual represents a point in search space and possible solution. Each individual is represented as a string of character/integer/float/bits. This string is analogous to the Chromosome.

To implement genetic algorithm we'll use TPOT which is an open-source library for performing AutoML in Python. It makes use of the popular Scikit-Learn machine learning library for data transforms and machine learning algorithms and uses a Genetic Programming stochastic global search procedure to efficiently discover a top-performing model pipeline for a given dataset.

We'll first have to numberize the training and test label set. Here we use `sklearn` 's `LabelEncoder` class to implement this.

In [216…
```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

y_train_n = label_encoder.fit_transform(y_train)
y_test_n = label_encoder.fit_transform(y_test)

y_train_n
```

Out[216…
```
array([3, 3, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 3, 2, 2, 1, 1, 0, 1, 0, 0,
       0, 0, 0, 3, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 2, 1, 2, 2, 0, 1, 1, 2,
       1, 1, 1, 2, 2, 1, 0, 2, 1, 1, 0, 2, 1, 2, 0, 0, 1, 1, 1, 1, 1, 3,
       1, 0, 0, 1, 1, 2, 2, 1, 1, 3, 1, 2, 1, 3, 2, 1, 1, 0, 1, 1, 0, 0,
       2, 0, 0, 0, 2, 2, 3, 0, 0, 1, 3, 1, 0, 1, 3, 1, 2, 0, 1, 3, 1, 0,
       1, 1, 1, 2, 2, 0, 0, 2, 1, 0, 2, 1, 2, 0, 1, 2, 1, 3, 1, 1, 1, 2,
       3, 1, 0, 3, 1, 1, 2, 2, 0, 1, 1, 2, 1, 3, 2, 2, 2, 3, 3, 2, 0, 1,
       1, 3, 1, 3, 2, 0, 2, 1, 0, 3, 0, 1, 2, 1, 0, 1, 2, 0, 2, 1, 2, 1,
       1, 2, 2, 0, 1, 1, 2, 0, 2, 1, 2, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 3, 3, 2, 0, 1, 1, 1, 1, 1, 1, 2, 0, 2, 1, 2, 2, 3, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 2, 1, 1, 0, 2, 2, 1, 2, 1,
       2, 0, 3, 1, 3, 2, 2, 2, 2, 2, 1, 1, 2, 1, 3, 1, 0, 2, 3, 2, 0, 1,
       1, 3, 0, 3, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 2,
       3, 2, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 2, 3, 2, 3, 1, 2, 0, 1, 0,
       0, 2, 2, 0, 0, 2, 0, 1, 1, 0, 0, 1, 2, 1, 0, 1, 2, 2, 1, 3, 1, 1,
       2, 1, 0, 2, 1, 2, 0, 1, 1, 3, 1, 1, 0, 1, 2, 1, 1, 1, 2, 1, 0, 1,
       2, 1, 0, 2, 0, 1, 0, 1, 2, 2, 1, 2, 2, 0, 1, 3, 2, 0, 0, 2, 3, 1,
       2, 0, 1, 2, 2, 3, 0, 1, 3, 0, 2, 0, 1, 3, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 2, 1, 0, 0, 2, 0, 3, 2, 1, 1, 2, 1, 3, 1, 0, 1, 2, 2, 1, 1, 2,
       2, 2, 0, 1, 1, 0, 0, 2, 2, 0, 2, 2, 2, 0, 0, 3, 2, 0, 1, 3, 2, 1,
       2, 0, 0, 1, 2, 0, 1, 2, 0, 1, 3, 2, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 3, 1, 0, 1, 3, 0, 1, 0, 1, 1, 0, 0, 0, 3, 1, 1, 1, 2, 0, 0, 1,
       0, 1, 1, 1, 0])
```

In [217…
```
y_train.head(20)
```

Out[217…
```
95      Not Satisfactory
63      Not Satisfactory
175            Excellent
71             Excellent
32             Excellent
64             Excellent
199            Excellent
197                 Good
284            Excellent
21                  Good
266            Excellent
3                   Good
27             Excellent
90      Not Satisfactory
13                  Good
205                 Good
73             Excellent
196            Excellent
96               Average
83             Excellent
Name: Academic Performance, dtype: object
```

Here we see our labels are encoded according to the following:

1. **Excellent** - **1**

1. **Good** - **2**

1. **Average** - **0**

1. **Not Satisfactory** - **3**

Let's finally train the Genetic Algorithm using `TPOTClassifier` . We are currently using 15 `generations` , 100 `population_size` and 150 `offspring_size` .

In [218…
```
from tpot import TPOTClassifier


tpot = TPOTClassifier(generations=15, population_size=100, offspring_size=150,
                      verbosity=2, early_stop=8, cv = 10, scoring = 'accuracy
                      random_state=42)

tpot.fit(X_train, y_train_n)
print(tpot.score(X_test, y_test_n))
tpot.export('tpot_digits_pipeline_comb.py')
```

```
Generation 1 - Current best internal CV score: 0.6891156462585033

Generation 2 - Current best internal CV score: 0.7076105442176871

Generation 3 - Current best internal CV score: 0.7076105442176871

Generation 4 - Current best internal CV score: 0.709608843537415

Generation 5 - Current best internal CV score: 0.709608843537415

Generation 6 - Current best internal CV score: 0.709608843537415
```

```
Generation 7 - Current best internal CV score: 0.7096938775510204

Generation 8 - Current best internal CV score: 0.7178146258503402

Generation 9 - Current best internal CV score: 0.7178146258503402

Generation 10 - Current best internal CV score: 0.7178146258503402

Generation 11 - Current best internal CV score: 0.7178571428571429

Generation 12 - Current best internal CV score: 0.7261054421768708

Generation 13 - Current best internal CV score: 0.7261054421768708

Generation 14 - Current best internal CV score: 0.7261054421768708

Generation 15 - Current best internal CV score: 0.7280612244897958

Best pipeline: KNeighborsClassifier(CombineDFs(RandomForestClassifier(input_ma
trix, bootstrap=True, criterion=gini, max_features=0.3, min_samples_leaf=1, mi
n_samples_split=7, n_estimators=100), input_matrix), n_neighbors=3, p=1, weigh
ts=distance)
0.7
```

Genetic algorithm showed us that the most optimized algorithm is the `KNeighborsClassifier` with the following parameter :

`KNeighborsClassifier(CombineDFs(RandomForestClassifier(input_matrix, bootstrap=True, criterion=gini, max_features=0.3, min_samples_leaf=1, min_samples_split=7, n_estimators=100), input_matrix), n_neighbors=3, p=1, weights=distance)`

`0.7`

Let's fit this algorithm to our dataset and check the training score

In [220…
```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline, make_union
from tpot.builtins import StackingEstimator
from tpot.export_utils import set_param_recursive
from sklearn.preprocessing import FunctionTransformer
from copy import copy


# Average CV score on the training set was: 0.7280612244897958
exported_pipeline = make_pipeline(
    make_union(
        StackingEstimator(estimator=RandomForestClassifier(bootstrap=True, cri
        FunctionTransformer(copy)
    ),
    KNeighborsClassifier(n_neighbors=3, p=1, weights="distance")
)
# Fix random state for all the steps in exported pipeline
set_param_recursive(exported_pipeline.steps, 'random_state', 42)

exported_pipeline.fit(X_train, y_train_n)
results = exported_pipeline.predict(X_test)
```

Let's check the accuracy on the test set and check the confusion matrix, precision, recall and f1 scores.

In [221…
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


conf_mat = confusion_matrix(y_test_n, results)
class_report = classification_report(y_test_n, results)


print("Accuracy:", metrics.accuracy_score(y_test_n, results))

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.7
[[37  5  3  6]
 [ 3 72  9  1]
 [ 7  9 28  2]
 [11  3  4 10]]
              precision    recall  f1-score   support

           0       0.64      0.73      0.68        51
           1       0.81      0.85      0.83        85
           2       0.64      0.61      0.62        46
           3       0.53      0.36      0.43        28

    accuracy                           0.70       210
   macro avg       0.65      0.63      0.64       210
weighted avg       0.69      0.70      0.69       210
```

Finally, let's perform `KFold` cross validation.

In [222…

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_ga = KFold(n_splits=10, shuffle=True, random_state=42)

scores = cross_val_score(exported_pipeline, X_train, y_train_n, cv=cv_ga, sco
print('Training Accuracy On KFold Cross Validation: %.3f (%.3f)' % (np.mean(s

scores = cross_val_score(exported_pipeline, X_test, y_test_n, cv=cv_ga, scori
print('Testing Accuracy On KFold Cross Validation: %.3f (%.3f)' % (np.mean(sc
```

```
Training Accuracy On KFold Cross Validation: 0.697 (0.058)
Testing Accuracy On KFold Cross Validation: 0.610 (0.102)
```

This model givess us a 61% accuracy on `KFold` cross validation.

In [ ]: