Analyzing The College Dataset

First let's import the necessary libraries.

In [1]:
```
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 [2]:
```
%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 [3]:
```
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 [4]:
```
college_df = load_the_dataset('COLLEGE_N.csv')
```

Let's check the data.

In [5]:
```
college_df.head()
```

Out[5]:

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Residenc
0	Female	17	Google	Very Good	Mobile	Home	Not Connected	Night	Daily	Towr
1	Female	17	Facebook	Good	Mobile	Home	Not Connected	Night	Daily	Towr

	Gender	Age	Popular Website	Proficiency	Medium	Location	Household Internet Facilities	Browse Time	Browsing Status	Residence
2	Female	17	Youtube	Very Good	Mobile	Home	Not Connected	Night	Daily	Towı

Check the dataset using `info()` .

In [6]:
```
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
```

Let's check the `shape` .

In [7]:
```
college_df.shape
```

Out[7]:
```
(199, 20)
```

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

In [8]:
```
college_df['Gender'].value_counts()
```

Out[8]:
```
Female    131
Male       68
```

```
Name: Gender, dtype: int64
```

Check Age

In [9]:
```
college_df['Age'].value_counts()
```

Out[9]:
```
17    171
18     12
16      8
15      8
Name: Age, dtype: int64
```

Check Frequently Visited Website

In [10]:
```
college_df['Popular Website'].value_counts()
```

Out[10]:
```
Google      54
Facebook    44
Whatsapp    43
Youtube     31
Gmail       18
Twitter      9
Name: Popular Website, dtype: int64
```

In [11]:
```
college_df.rename(columns={
    'Popular Website':'Frequently Visited Website',
}, inplace=True)

college_df.columns
```

Out[11]:
```
Index(['Gender', 'Age', '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', 'Priority of Learning',
       'Webinar', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

Check Effectiveness Of Internet Usage

In [12]:
```
college_df['Proficiency'].value_counts()
```

Out[12]:
```
Very Good    71
Good         69
Average      59
Name: Proficiency, dtype: int64
```

In [13]:
```
college_df.rename(columns={
    'Proficiency':'Effectiveness Of Internet Usage'
}, inplace=True)

college_df.columns
```

Out[13]:
```
Index(['Gender', 'Age', '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', 'Priority of Learning',
       'Webinar', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
```

In [14]:
```
college_df.replace({'Effectiveness Of Internet Usage': {'Very Good':'Very Effe
                                                        'Average':'Somewhat E
```

In [15]:
```
college_df['Effectiveness Of Internet Usage'].value_counts()
```

Out[15]:
```
Very Effective        71
Effective             69
Somewhat Effective    59
Name: Effectiveness Of Internet Usage, dtype: int64
```

Check Devices Used For Internet Browsing

In [16]:
```
college_df['Medium'].value_counts()
```

Out[16]:
```
Mobile               159
Laptop and Mobile     27
Desktop               13
Name: Medium, dtype: int64
```

In [17]:
```
college_df.rename(columns={
    'Medium':'Devices Used For Internet Browsing',
}, inplace=True)

college_df.columns
```

Out[17]:
```
Index(['Gender', 'Age', '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', 'Priority of Learning',
       'Webinar', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

Check Location Of Internet Use

In [18]:
```
college_df['Location'].value_counts()
```

Out[18]:
```
Home          186
College        12
Cyber Cafe      1
Name: Location, dtype: int64
```

In [19]:
```
college_df.rename(columns={
    'Location':'Location Of Internet Use'
}, inplace=True)

college_df.columns
```

Out[19]:
```
Index(['Gender', 'Age', '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',
       'Priority of Learning', 'Webinar',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Household Internet Facilities

In [20]:
```python
college_df['Household Internet Facilities'].value_counts()
```

Out[20]:
```
Not Connected    176
Connected         23
Name: Household Internet Facilities, dtype: int64
```

Check Time Of Internet Browsing

In [21]:
```python
college_df['Browse Time'].value_counts()
```

Out[21]:
```
Night       168
Day          30
Midnight      1
Name: Browse Time, dtype: int64
```

In [22]:
```python
college_df.rename(columns={
       'Browse Time':'Time Of Internet Browsing',
}, inplace=True)

college_df.columns
```

Out[22]:
```
Index(['Gender', 'Age', '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',
       'Priority of Learning', 'Webinar',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Frequency Of Internet Usage

In [23]:
```python
college_df['Browsing Status'].value_counts()
```

Out[23]:
```
Daily      156
Weekly      40
Monthly      3
Name: Browsing Status, dtype: int64
```

In [24]:
```python
college_df.rename(columns={
    'Browsing Status':'Frequency Of Internet Usage',
}, inplace=True)

college_df.columns
```

Out[24]:
```
Index(['Gender', 'Age', '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',
        'Priority of Learning', 'Webinar',
        'Internet Usage For Educational Purpose', 'Academic Performance',
        'Obstacles'],
```

Check Place Of Student's Residence

In [25]:
```
college_df['Residence'].value_counts()
```

Out[25]:
```
Town       167
Village     25
Remote       7
Name: Residence, dtype: int64
```

In [26]:
```
college_df.rename(columns={
    'Residence':'Place Of Student\'s Residence',
}, inplace=True)

college_df.columns
```

Out[26]:
```
Index(['Gender', 'Age', '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', 'Priority of Learning',
       'Webinar', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
      dtype='object')
```

Check Purpose Of Internet Use

In [27]:
```
college_df['Purpose of Use'].value_counts()
```

Out[27]:
```
Social Media       67
Entertainment      45
Education          27
Blog               22
News               20
Online Shopping    18
Name: Purpose of Use, dtype: int64
```

In [28]:
```
college_df.rename(columns={
    'Purpose of Use':'Purpose Of Internet Use',
}, inplace=True)

college_df.columns
```

Out[28]:
```
Index(['Gender', 'Age', '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', 'Priority of Learning',
       'Webinar', 'Internet Usage For Educational Purpose',
       'Academic Performance', 'Obstacles'],
```

Check Browsing Purpose

In [29]:
```python
college_df['Browsing Purpose'].value_counts()
```

Out[29]:
```
Non-academic    115
Academic         84
Name: Browsing Purpose, dtype: int64
```

Check Webinar

In [30]:
```python
college_df['Webinar'].value_counts()
```

Out[30]:
```
No     171
Yes     28
Name: Webinar, dtype: int64
```

Check Priority Of Learning On The Internet

In [31]:
```python
college_df['Priority of Learning'].value_counts()
```

Out[31]:
```
Non-academic Learning              58
Communication Skills               44
Academic Learning                  39
Creativity and Innovative Skills   24
Leadership Development             19
Career Opportunity                 15
Name: Priority of Learning, dtype: int64
```

In [32]:
```python
college_df.rename(columns={
    'Priority of Learning':'Priority Of Learning On The Internet',
}, inplace=True)

college_df.columns
```

Out[32]:
```
Index(['Gender', 'Age', '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',
       'Priority Of Learning On The Internet', 'Webinar',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Obstacles'],
      dtype='object')
```

Check Internet Usage For Educational Purpose

In [33]:
```python
college_df['Internet Usage For Educational Purpose'].value_counts()
```

Out[33]:
```
Articles or Blogs related to non-academical studies    59
```

```
Notes or lectures for academical purpose            45
Articles or Blogs related to academical studies     37
E-books or other Media files                        33
Courses Available on specific topics                25
```

Check Academic Performance

In [34]:
```
college_df['Academic Performance'].value_counts()
```

Out[34]:
```
Average             91
Satisfactory        44
Not Satisfactory    38
Good                26
Name: Academic Performance, dtype: int64
```

In [35]:
```
college_df.replace({'Academic Performance': {'Good':'Excellent', 'Satisfactory
```

In [36]:
```
college_df['Academic Performance'].value_counts()
```

Out[36]:
```
Average             91
Good                44
Not Satisfactory    38
Excellent           26
Name: Academic Performance, dtype: int64
```

Check Barriers To Internet Access

In [37]:
```
college_df['Obstacles'].value_counts()
```

Out[37]:
```
Bad Service       88
High Price        83
Unavailability    28
Name: Obstacles, dtype: int64
```

In [38]:
```
college_df.rename(columns={
    'Obstacles':'Barriers To Internet Access',
}, inplace=True)

college_df.columns
```

Out[38]:
```
Index(['Gender', 'Age', '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',
       'Priority Of Learning On The Internet', 'Webinar',
       '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 [39]:
```
# Write a function to save the figures
PROJECT_ROOT_DIR = "."
DATASET_ID = "College"
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 [40]:
```
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 [41]:
```
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 [42]:
```
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'].inde
             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', for
```

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 [43]:
```
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)
```

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

These are helper functions.

In [44]:
```
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 [45]:
```
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'] == 'A
    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_na

    return dictionary
```

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

In [46]:
```
def cat_vs_cat_bar_plot_browsing_purpose(dataframe, column_name, column_cat_l
    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, d
    dictionary = append_dataframe_to_dict(non_academic_df, column_name, labels

    return dictionary
```

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

In [47]:
```
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

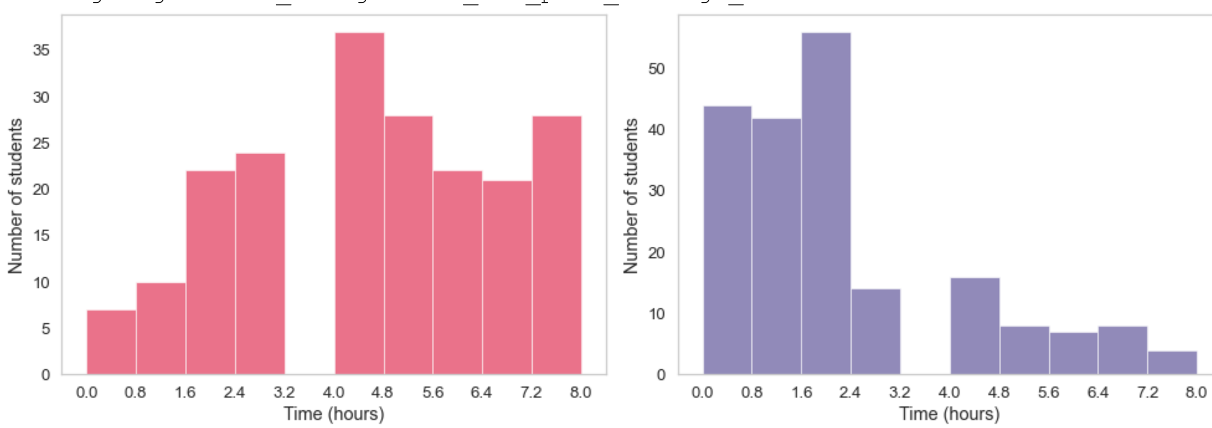
Only `'Total Internet Usage(hrs/day)'`, `'Time Spent in Academic(hrs/day)'`, `'Duration Of Internet Usage(In Years)'` are the non-categorical values in the dataset.

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

In [48]:

```
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(college_df['Total Internet Usage(hrs/day)'], 'Total_Intern
                    title = 'Total internet usage in a day',
                    xlabel = 'Time (hours)', ylabel = 'Number of students')

plt.subplot(122)
numerical_data_plot(college_df['Time Spent in Academic(hrs/day)'], 'Time_Spent
                    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



In [49]:

```
# 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)

# numerical_data_plot(college_df['Duration Of Internet Usage(In Years)'], 'Du
#                     hist_alpha=0.6, color='salmon', title='How Long Have The
#                     xlabel='Time(years)', ylabel='Number of Students')


# save_fig('Non_Categorical_Bar_plot_2')

# plt.show()
```

Plotting `Total Internet Usage(hrs/day)`

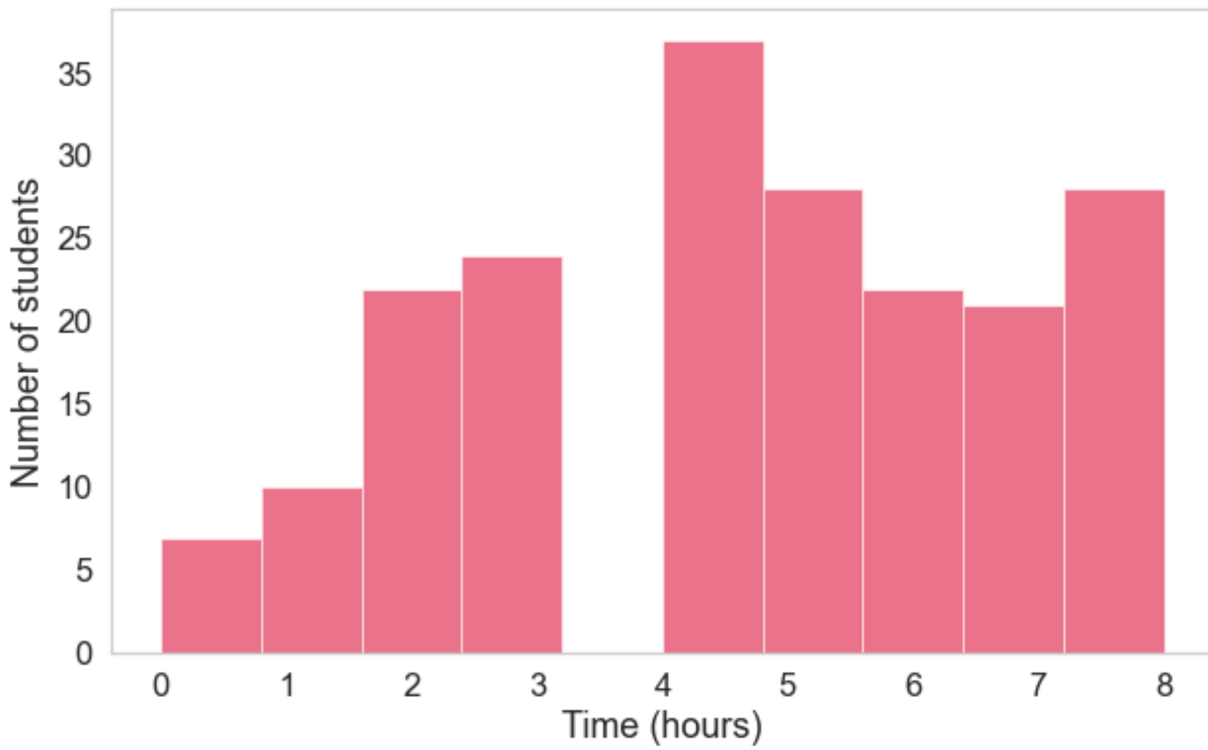
In [50]:

```
college_df['Total Internet Usage(hrs/day)'].value_counts()
```

Out[50]:
```
4    37
8    28
5    28
3    24
6    22
2    22
7    21
1    10
0     7
Name: Total Internet Usage(hrs/day), dtype: int64
```

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

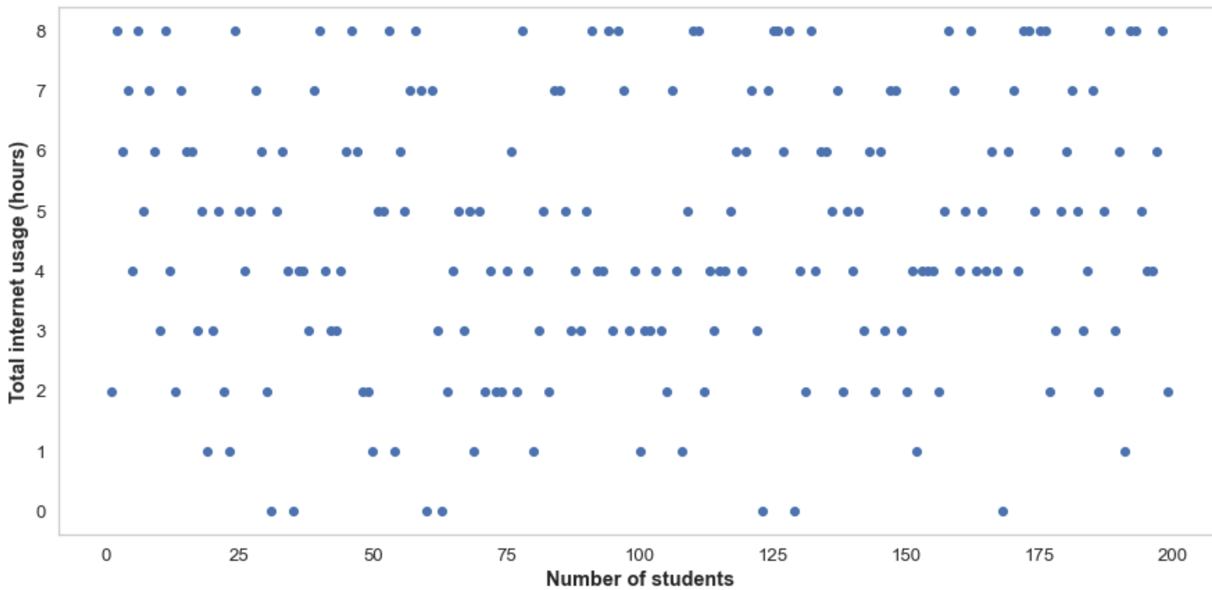
In [51]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

college_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 [52]:

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

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

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


plt.show()
```

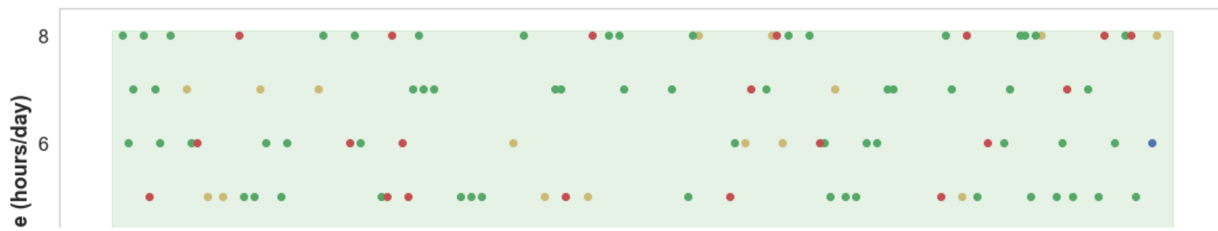


Now let's try plotting `Total Internet Usage(hrs/day)` against the target column `'Academic Performance'`.

In [53]:

```
categorical_scatter_plot(college_df, 'Total Internet Usage(hrs/day)', 'Academi
                        'Total Internet Usage In A Day W.R.T. Academic Perfor
                         'Total internet usage (hours/day)')

plt.fill_between([-1, 200], [3.2, 3.2], -0.2, color='steelblue', alpha=0.1, i
plt.fill_between([-1, 200], [8.1, 8.1], 3.3, color='green', alpha=0.1, interp

save_fig('Total_Internet_Usage_In_A_Day_WRT_Academic_Performance_Scatter_Plot

plt.show()
```

```
Saving figure Total_Internet_Usage_In_A_Day_WRT_Academic_Performance_Scatter_P
lot
```



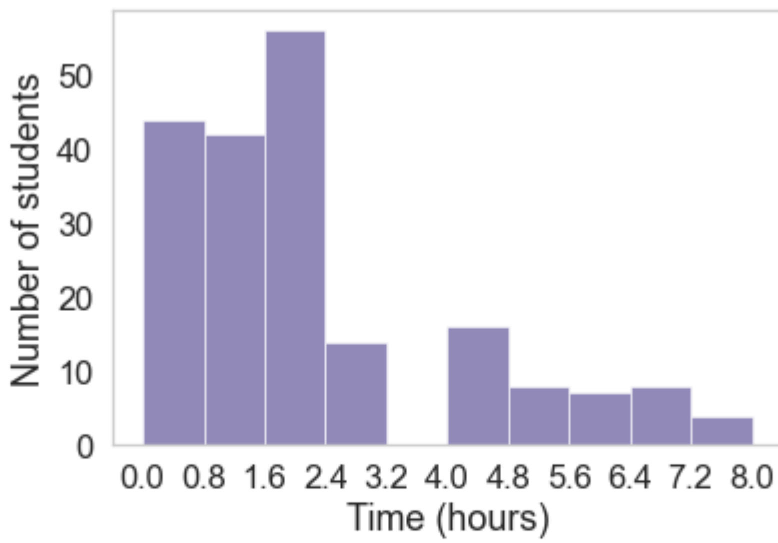
Plotting Time Spent in Academic(hrs/day)

In [54]:
```
college_df['Time Spent in Academic(hrs/day)'].value_counts()
```

Out[54]:
```
2    56
0    44
1    42
4    16
3    14
7     8
5     8
6     7
8     4
Name: Time Spent in Academic(hrs/day), dtype: int64
```

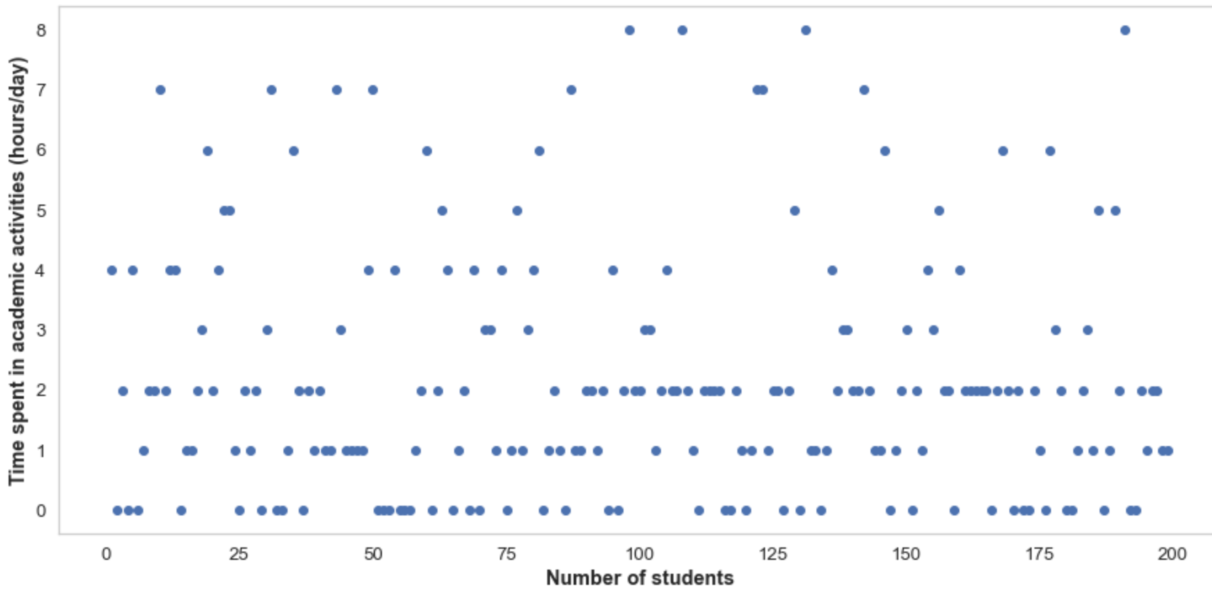
First let's check the histogram and the boxplot of this column.

In [55]:
```
numerical_data_plot(college_df['Time Spent in Academic(hrs/day)'], 'Time_Spent
                    hist_alpha=0.6, color='darkslateblue',
                    title='Total time spent in academic studies in a day',
                    xlabel='Time (hours)', ylabel='Number of students')

plt.show()
```



Now let's check the scatter plot.

In [56]:
```
plt.figure(figsize=(15,7))
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(np.linspace(1, len(college_df.index), len(college_df.index)),
         college_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 [57]:
```
categorical_scatter_plot(college_df, 'Time Spent in Academic(hrs/day)', 'Acade
                         'Time Spent In Academic In A Day W.R.T. Academic Per
                         'Time spent in academic activities (hours/day)')

plt.fill_between([-1, 200], [8.2, 8.2], 3.8, color='steelblue', alpha=0.1, int
plt.fill_between([-1, 200], [3.7, 3.7], -0.2, color='green', alpha=0.1)

save_fig('Time_Spent_In_Academic_In_A_Day_WRT_Academic_Performance_Scatter_Plo

plt.show()
```

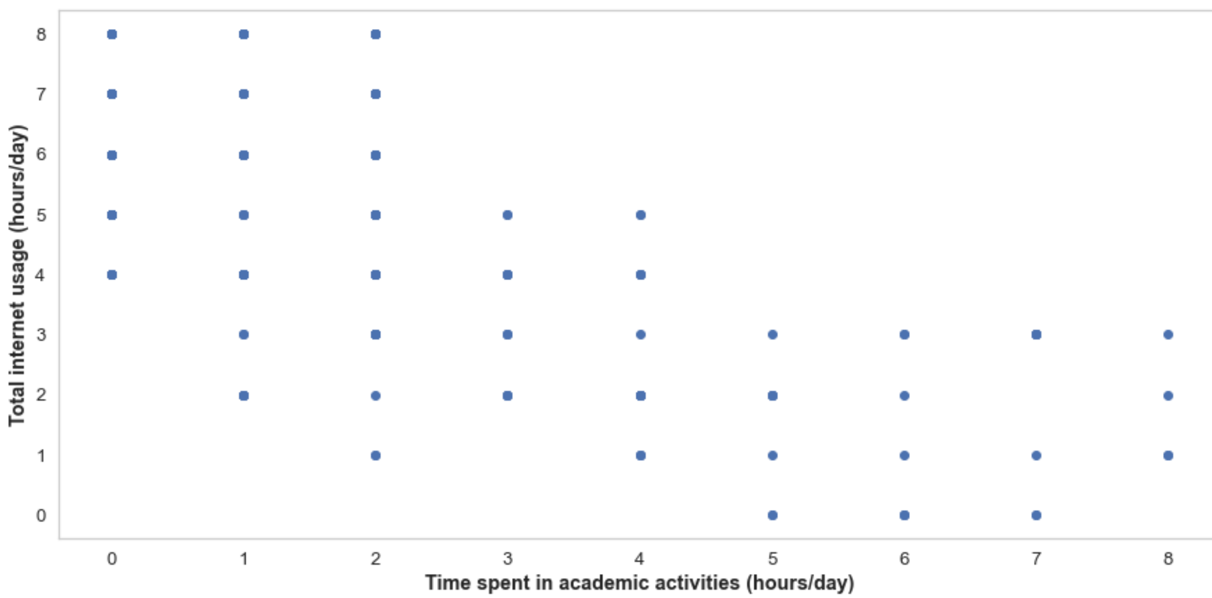
```
Saving figure Time_Spent_In_Academic_In_A_Day_WRT_Academic_Performance_Scatter
_Plot
```

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

Let's use scatter plot.

In [58]:

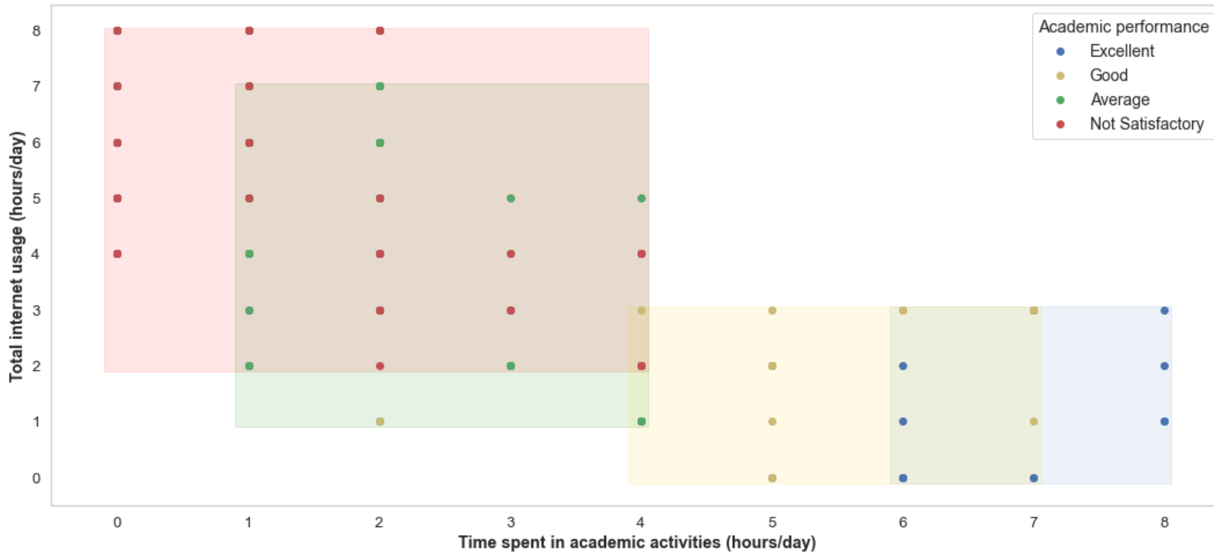
```
plt.figure(figsize=(15, 7))
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(college_df['Time Spent in Academic(hrs/day)'],
         college_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 [59]:
```
categorical_scatter_plot_wrt_academic_performance(college_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([5.9, 8.05], [3.05, 3.05], -0.1, color='steelblue', alpha=0.1
plt.fill_between([3.9, 7.05], [3.05, 3.05], -0.1, color='gold', alpha=0.1, int

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

```
Saving figure Time_Spent_in_Academic_vs_Total_Internet_Usage_Scatter_Plot
```



Plotting Duration Of Internet Usage(In Years)

In [60]:
```
college_df.rename(columns={
    'Years of Internet Use':'Duration Of Internet Usage(In Years)',
}, inplace=True)

college_df.columns
```

Out[60]:
```
Index(['Gender', 'Age', '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',
       'Priority Of Learning On The Internet', 'Webinar',
       'Internet Usage For Educational Purpose', 'Academic Performance',
       'Barriers To Internet Access'],
      dtype='object')
```

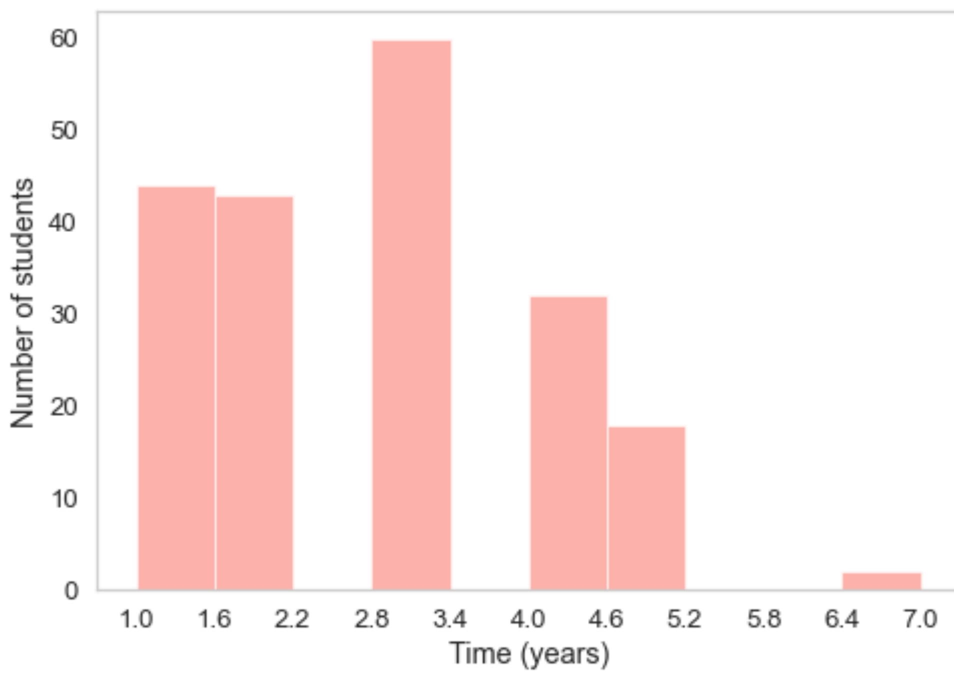
In [61]:
```
college_df['Duration Of Internet Usage(In Years)'].value_counts()
```

Out[61]:
```
3    60
1    44
2    43
4    32
5    18
7     2
Name: Duration Of Internet Usage(In Years), dtype: int64
```

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

In [62]:
```
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(college_df['Duration Of Internet Usage(In Years)'], 'Durat
                    hist_alpha=0.6, color='salmon',
                    title='How long have the students been using internet?', :
                    ylabel='Number of students')


save_fig('Non_Categorical_Bar_plot_2')

plt.show()
```

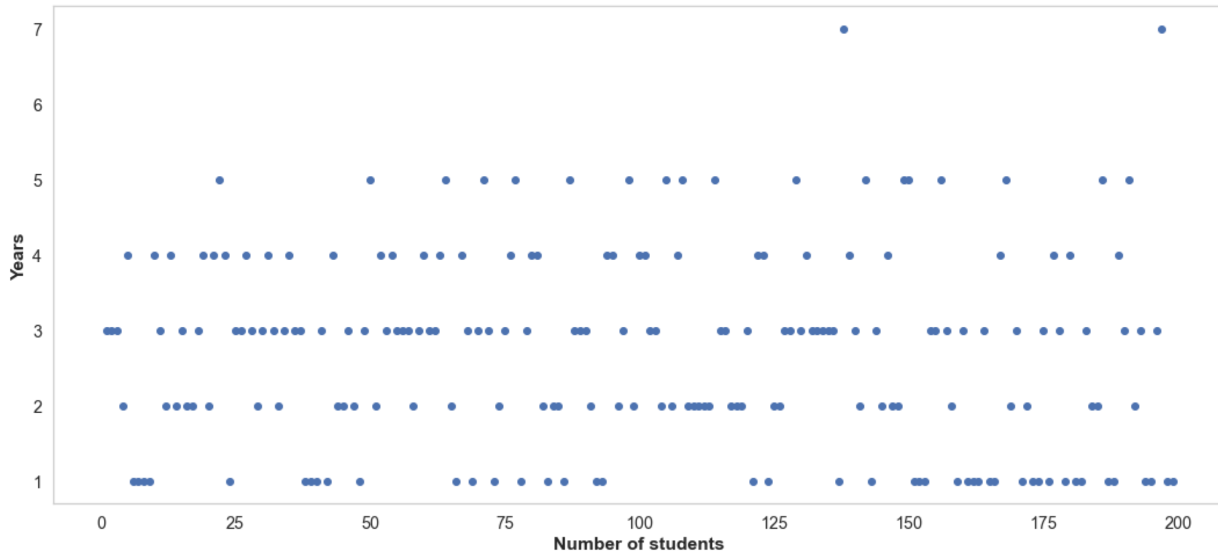
```
Saving figure Non_Categorical_Bar_plot_2
```



Now let's check the scatter plot.

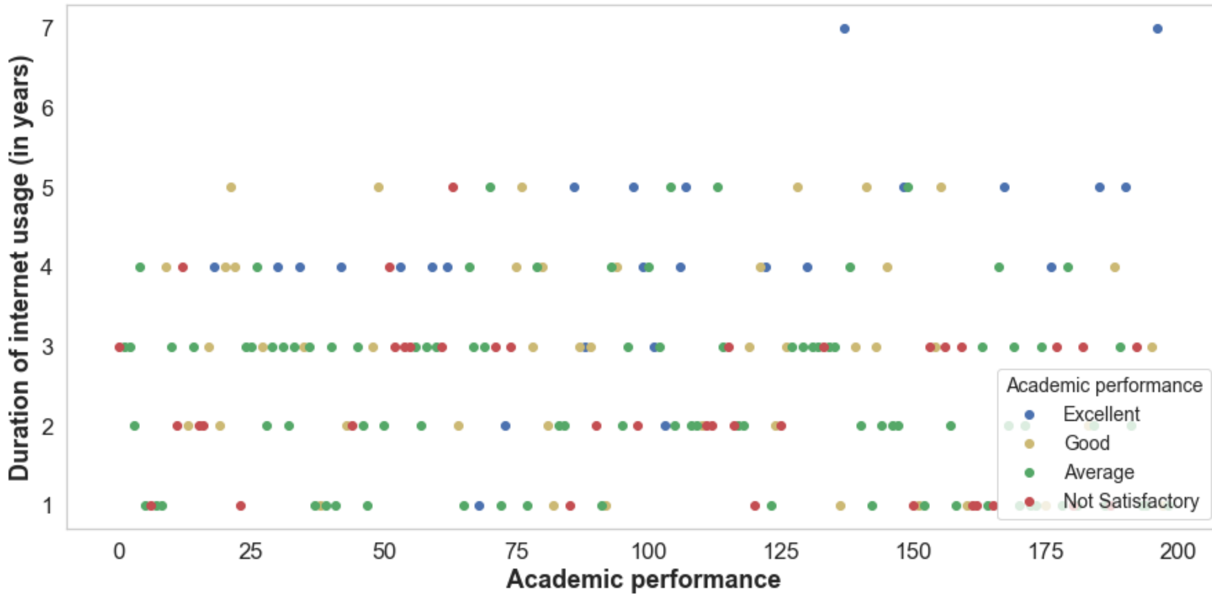
In [63]:
```
plt.figure(figsize=(15, 7))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.plot(np.linspace(1, len(college_df.index), len(college_df.index)),
         college_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
```



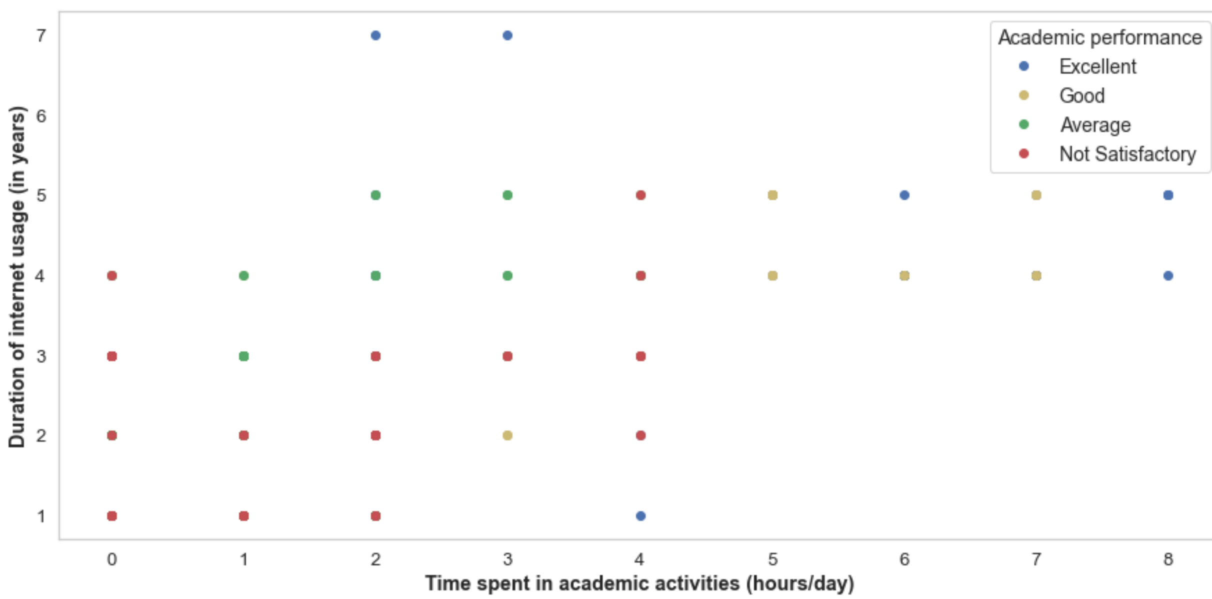
Now let's try plotting `'Years of Internet Use'` against the target column `'Academic Performance'`.

In [64]:
```
categorical_scatter_plot(college_df, 'Duration Of Internet Usage(In Years)',
                         'Duration Of Internet Usage(In Years) vs Academic Pe
                         'Duration of internet usage (in years)', 'Academic pe
```



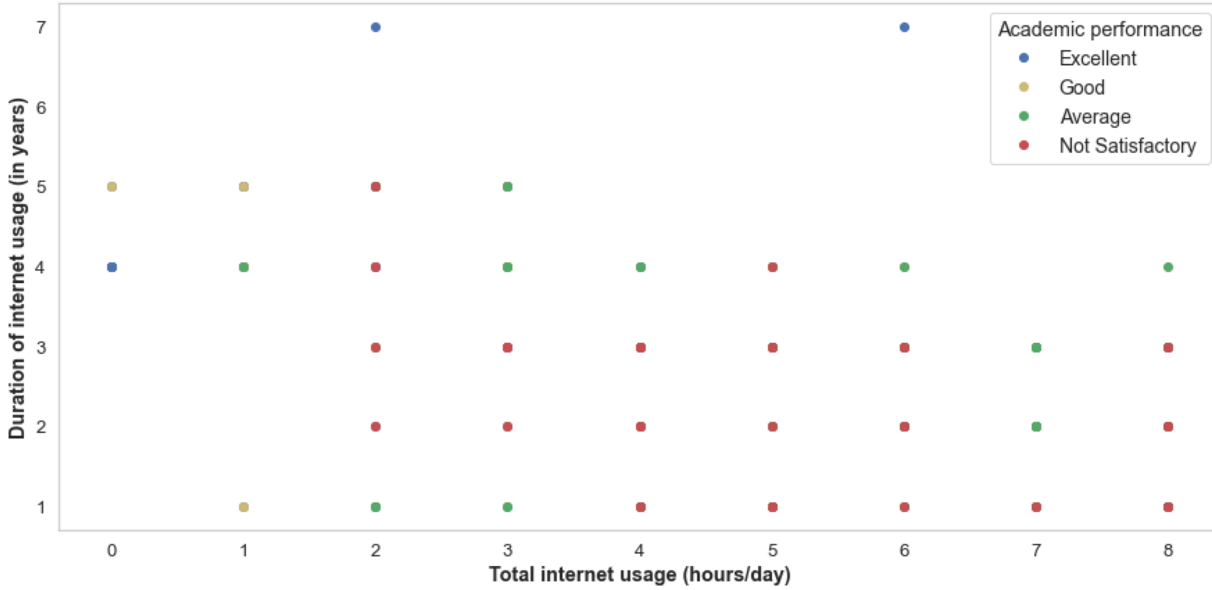
Now let's try plotting `Time Spent in Academic(hrs/day)` **vs** `'Duration Of Internet Usage(In Years)'` **against the target** `'Academic Performance'` **.**

In [65]:
```
categorical_scatter_plot_wrt_academic_performance(college_df, 'Time Spent in A
                                                  '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 [66]:
```
categorical_scatter_plot_wrt_academic_performance(college_df, 'Total Internet
                                                  'Duration Of Internet Usage
                                                  'Total Internet Usage (hrs/c
                                                  'Duration of internet usage
                                                  'Total internet usage (hours

plt.show()
```



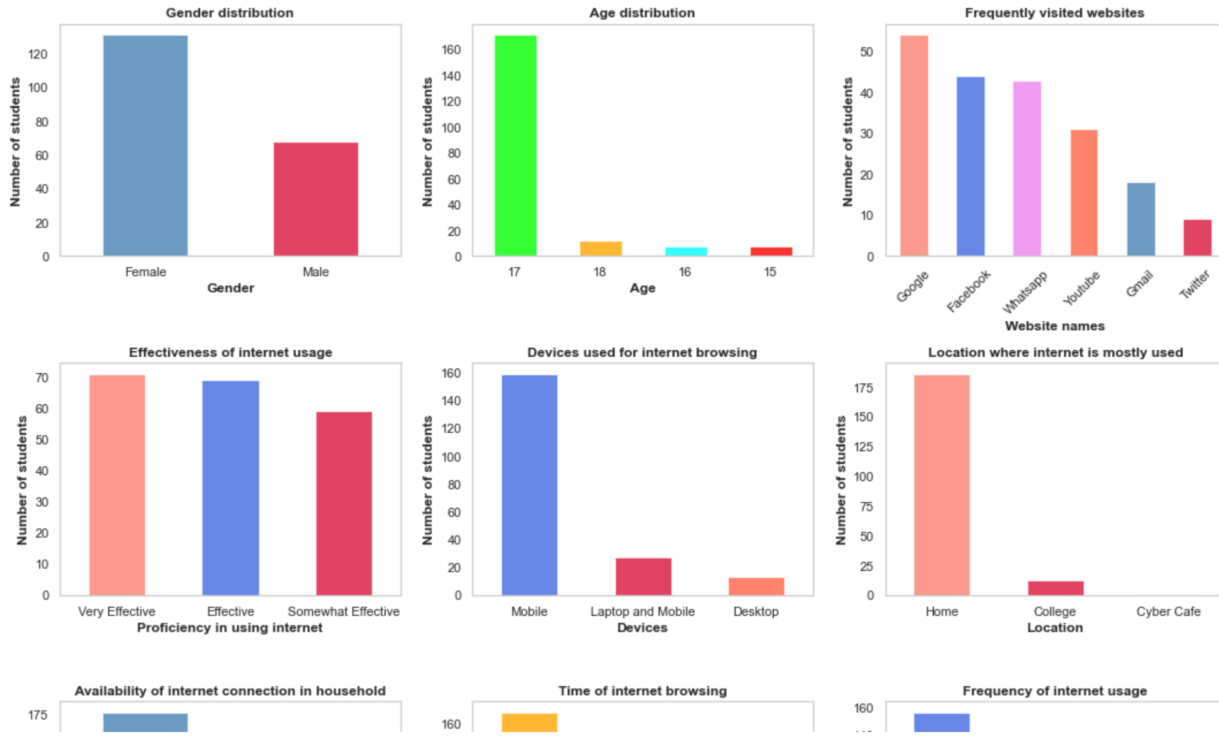
Plotting Categorical Values

'Gender' , 'Age' , '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.

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

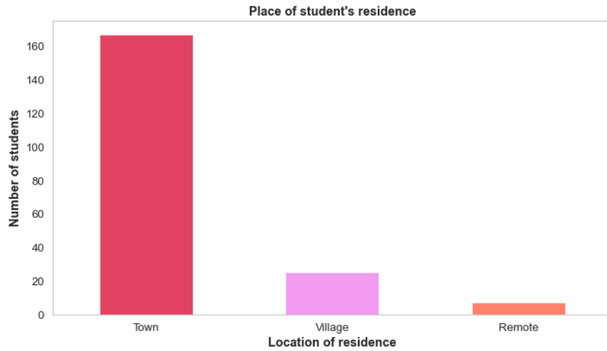
In [67]:
```
plt.figure(figsize=(15, 12))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1)
sns.set_style("whitegrid", {'axes.grid' : False})

plt.subplot(331)
categorical_bar_plot(college_df['Gender'], title='Gender distribution', xlabel

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

plt.subplot(333)
categorical_bar_plot(college_df['Frequently Visited Website'], rot=45,
                     color=['salmon', 'royalblue', 'violet', 'tomato', 'steelb
                     title='Frequently visited websites', xlabel='Website name

plt.subplot(334)
categorical_bar_plot(college_df['Effectiveness Of Internet Usage'], color=['sa
                     title='Effectiveness of internet usage', xlabel='Proficie

plt.subplot(335)
categorical_bar_plot(college_df['Devices Used For Internet Browsing'],
                     color=['royalblue', 'crimson', 'tomato', 'orange'],
                     title='Devices used for internet browsing', xlabel='Devic

plt.subplot(336)
categorical_bar_plot(college_df['Location Of Internet Use'],
                     color=['salmon', 'crimson', 'violet', 'orange', 'steelblu
                     title='Location where internet is mostly used', xlabel='I

plt.subplot(337)
categorical_bar_plot(college_df['Household Internet Facilities'],
                     title='Availability of internet connection in household',
                     xlabel='Household internet facilities')

plt.subplot(338)
categorical_bar_plot(college_df['Time Of Internet Browsing'], color=['orange',
                     title='Time of internet browsing', xlabel='Browsing time

plt.subplot(339)
categorical_bar_plot(college_df['Frequency Of Internet Usage'], color=['royalb
                     title='Frequency of internet usage', xlabel='Browsing sta


save_fig('Bar_plot_collage_1')

plt.show()
```

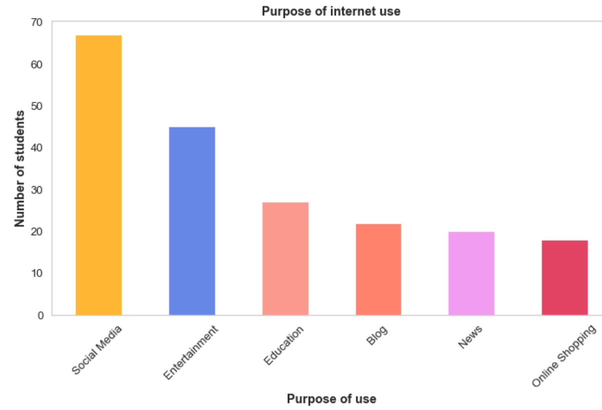
```
Saving figure Bar_plot_collage_1
```

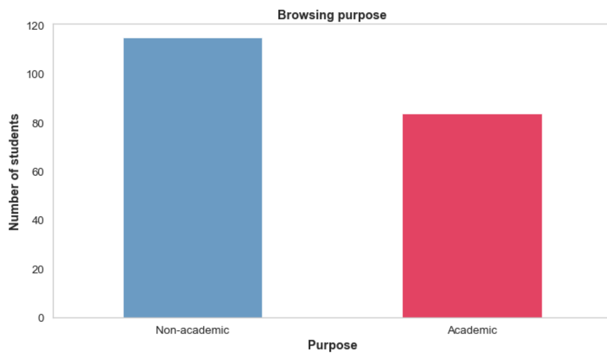

Gender distribution
Number of students
Female
Male
Gender
Age distribution
Number of students
17
18
16
15
Age
Frequently visited websites
Number of students
Google
Facebook
Whatsapp
Youtube
Gmail
Twitter
Website names
Effectiveness of internet usage
Number of students
Very Effective
Effective
Somewhat Effective
Proficiency in using internet
Devices used for internet browsing
Number of students
Mobile
Laptop and Mobile
Desktop
Devices
Location where internet is mostly used
Number of students
Home
College
Cyber Cafe
Location
Availability of internet connection in household
Time of internet browsing
Frequency of internet usage

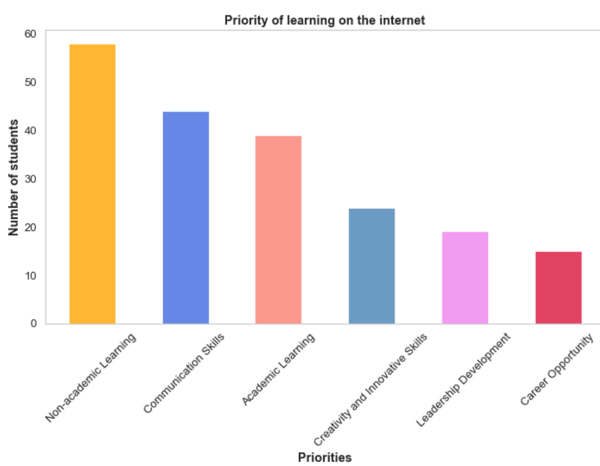
In [68]:
```
plt.figure(figsize=(20, 35))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})


plt.subplot(421)
categorical_bar_plot(college_df['Place Of Student\'s Residence'], color=['cri
                     title='Place of student\'s residence', xlabel='Location 

plt.subplot(422)
categorical_bar_plot(college_df['Purpose Of Internet Use'], rot=45,
                     color = ['orange', 'royalblue', 'salmon', 'tomato', 'vio
                     title='Purpose of internet use', xlabel='Purpose of use'

plt.subplot(423)
categorical_bar_plot(college_df['Browsing Purpose'], title='Browsing purpose',
                     xlabel='Purpose')

plt.subplot(424)
categorical_bar_plot(college_df['Webinar'], color=['salmon', 'crimson'],
                     title='Participation in webinars', xlabel='Participation 

plt.subplot(425)
categorical_bar_plot(college_df['Priority Of Learning On The Internet'], rot=
                     color = ['orange', 'royalblue', 'salmon', 'steelblue', '
                     title='Priority of learning on the internet', xlabel='Pr

plt.subplot(426)
categorical_bar_plot(college_df['Internet Usage For Educational Purpose'], rot
                     color=['orange', 'royalblue', 'salmon', 'steelblue', 'vi
                     title='Different reasons for internet browsing for educat
                     xlabel='Internet usage for educational purpose')

plt.subplot(427)
categorical_bar_plot(college_df['Academic Performance'], color=['salmon', 'ste
                     title='Academic performance', xlabel='Performance')

plt.subplot(428)
categorical_bar_plot(college_df['Barriers To Internet Access'],
                     color=['royalblue', 'darkslateblue', 'coral', 'crimson'],
                     title='Barriers to internet access', xlabel='Obstacles')


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

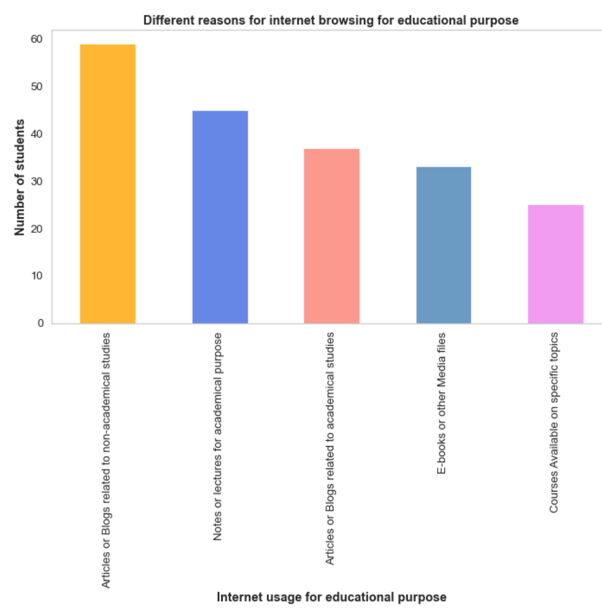
```
Saving figure Bar_plot_collage_2
```







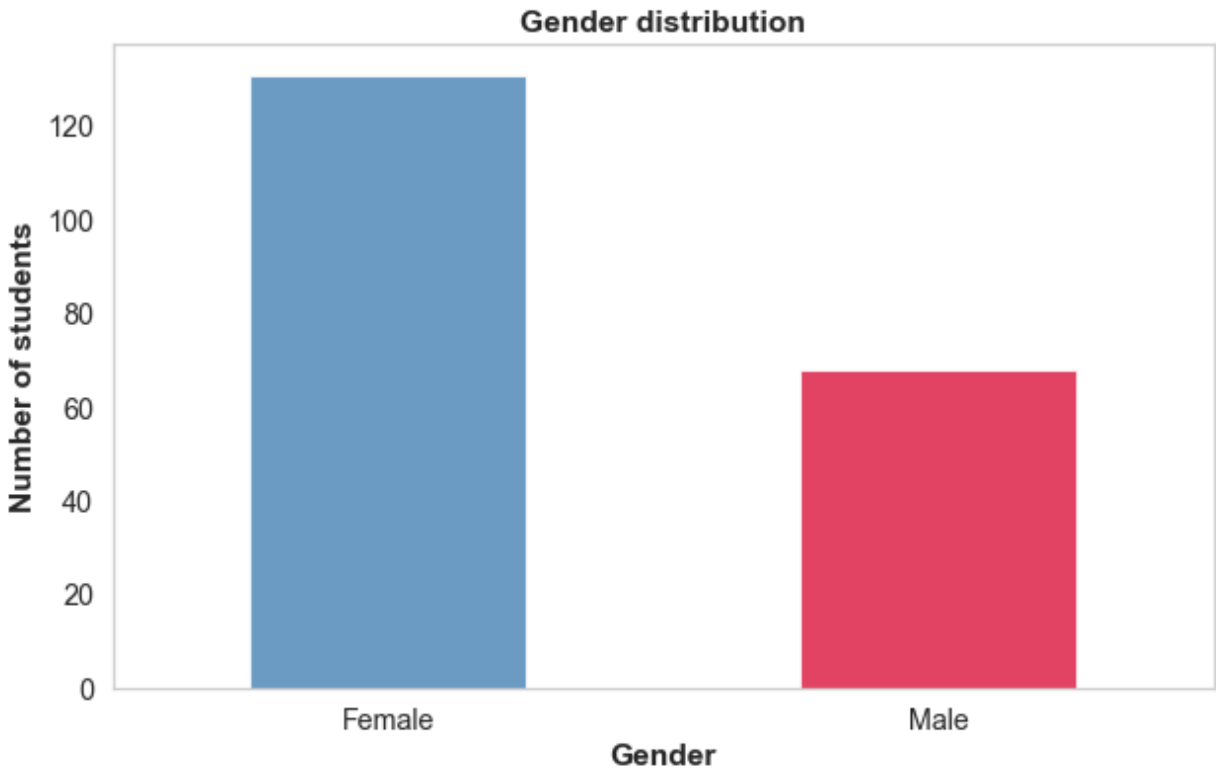




Plotting 'Gender'

Let's check the histogram.

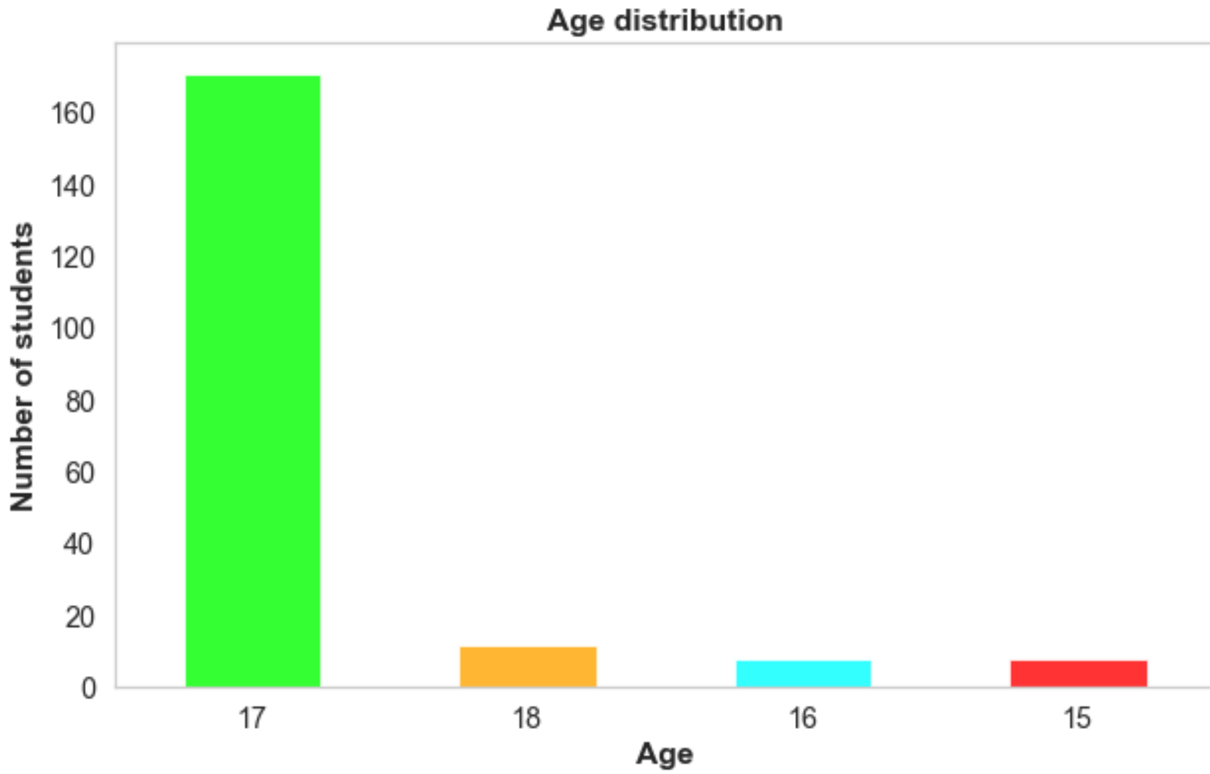
In [69]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_df['Gender'], title='Gender distribution', xlabel

plt.show()
```



Plotting 'Age'

Let's check the histogram.

In [70]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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 [71]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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 [72]:

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

dictionary = cat_vs_cat_bar_plot(college_df, 'Frequently Visited Website',
                                 college_df['Frequently Visited Website'].value_c

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 = 'G
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 W.R.T. Academic Performance', font
ax.set_xticks(x - width/3)
ax.set_xticklabels(labels)
ax.legend(title='Frequently visited websites', title_fontsize=14)

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_Histogram')
plt.show()
```

```
Saving figure Frequently_Visited_Websites_WRT_Academic_Performance_Histogram
```

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

In [73]:

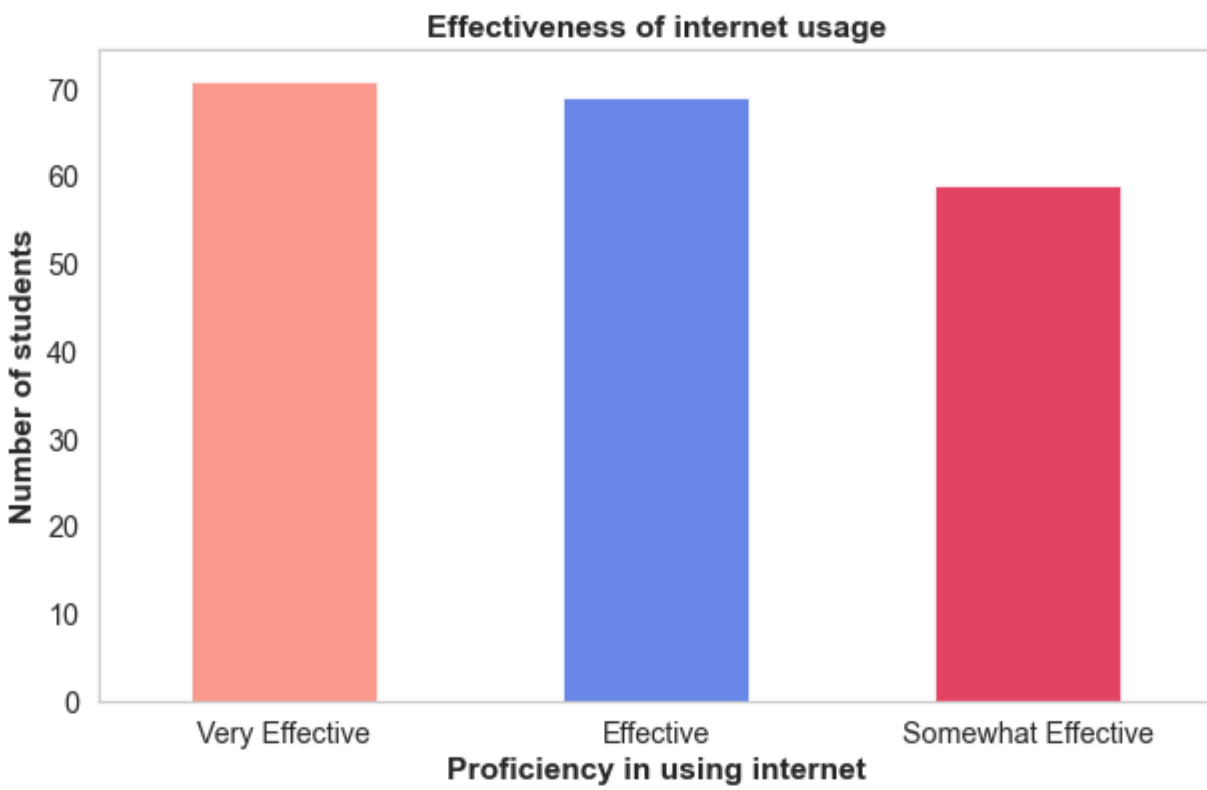
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_browsing_purpose(college_df, 'Frequently Vis
                              college_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, dictionary['Google'], width/2, label = 'Google')
rects2 = ax.bar(x - width/2, dictionary['Youtube'], width/2, label = 'Youtube
rects3 = ax.bar(x, dictionary['Facebook'], width/2, label = 'Facebook')
rects4 = ax.bar(x + width/2, dictionary['Whatsapp'], width/2, label = 'Whatsa

ax.set_ylabel('Number of students', fontweight = 'bold')
ax.set_xlabel('Browsing purpose', fontweight = 'bold')
# ax.set_title('Frequently Visited Websites W.R.T. Browsing Purpose', fontwei
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Frequently visited websites', title_fontsize=14 ,loc='upper 

sns.set(font_scale=1.2)

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

fig.tight_layout()

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

```
Saving figure Frequently_Visited_Websites_WRT_Browsing_Purpose_Histogram
```

Plotting 'Effectiveness Of Internet Usage'

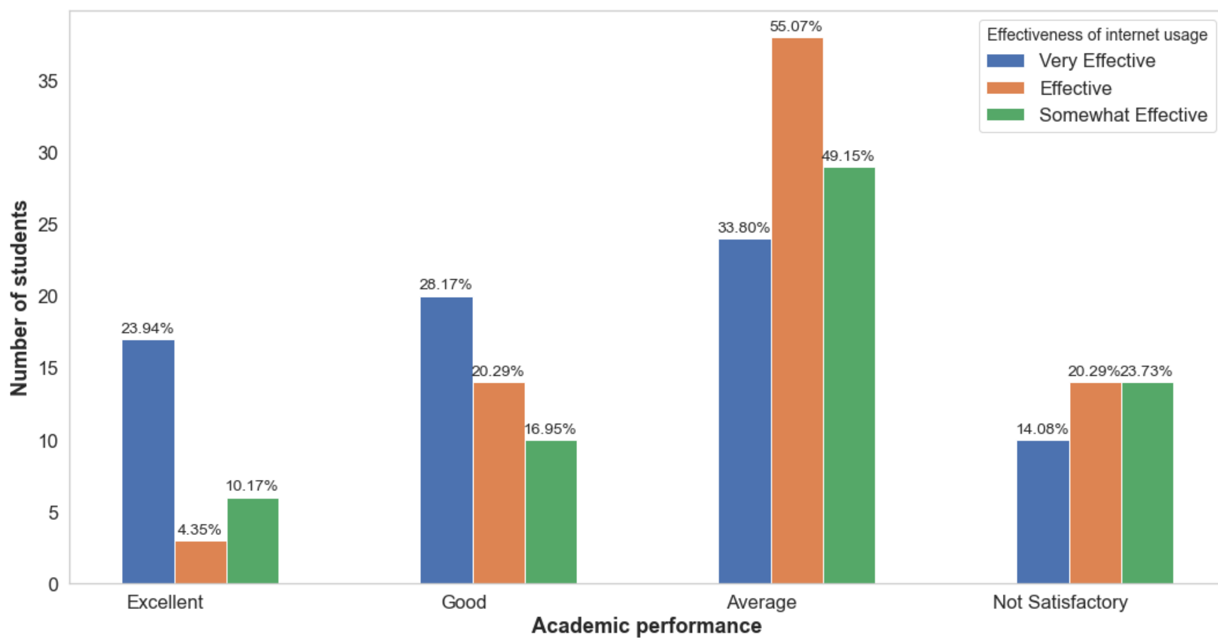
Let's check the histogram.

In [74]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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'` .

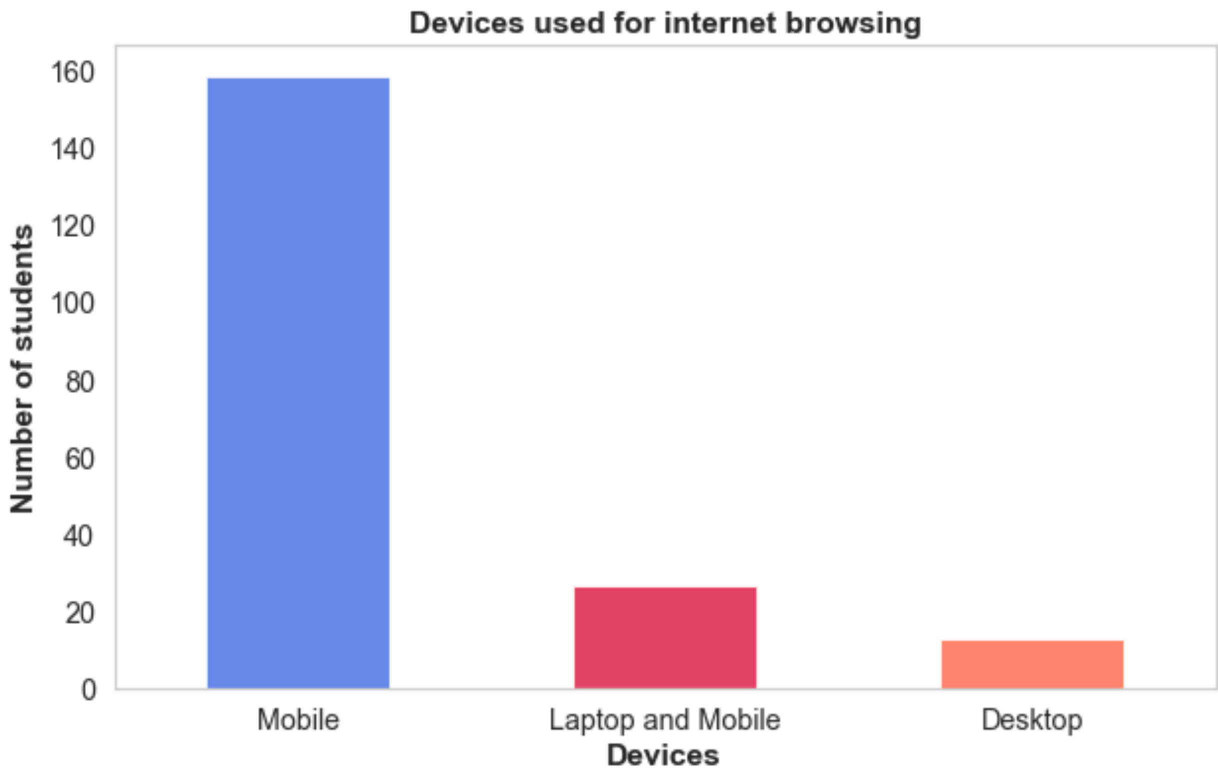
In [75]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_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/2, dictionary['Very Effective'], width/2, label = 'V
rects2 = ax.bar(x, dictionary['Effective'], width/2, label = 'Effective')
rects3 = ax.bar(x + width/2, dictionary['Somewhat Effective'], width/2, label

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=1.15)

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

fig.tight_layout()

plt.show()
```



Plotting 'Devices Used For Internet Browsing'

Let's check the histogram.

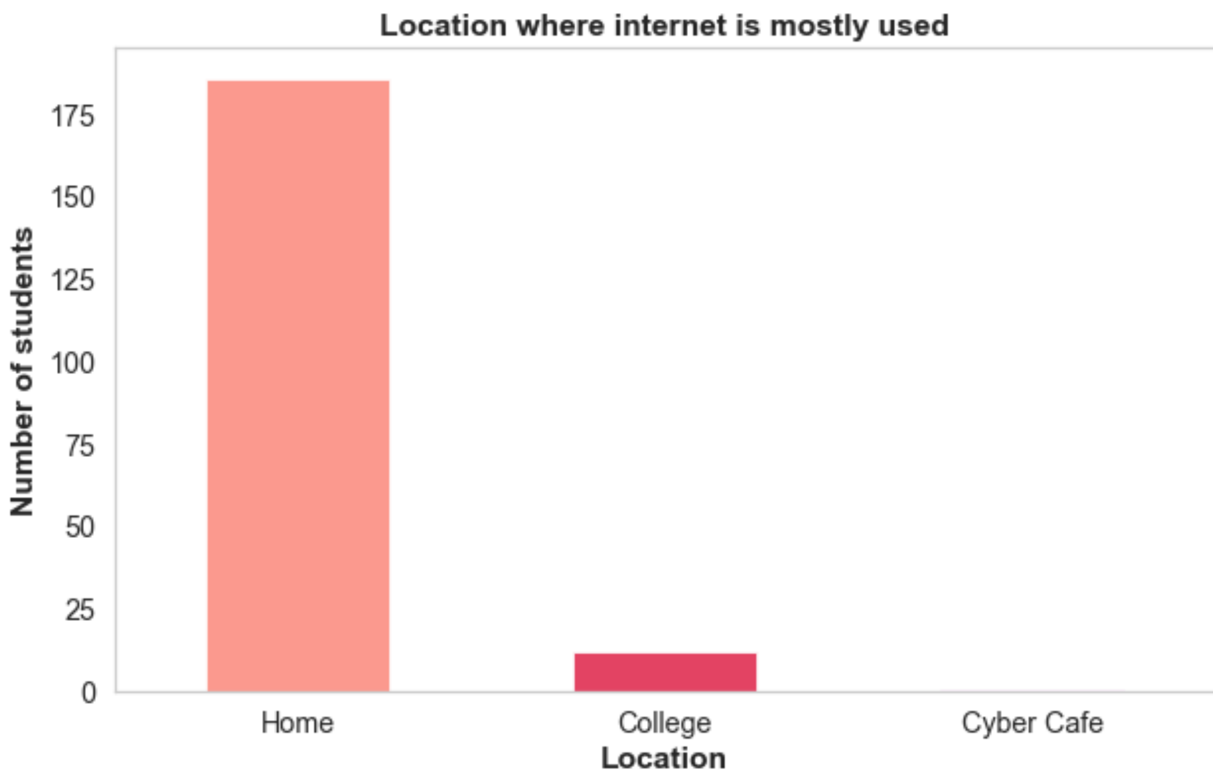
In [76]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```



Plotting 'Location Of Internet Use'

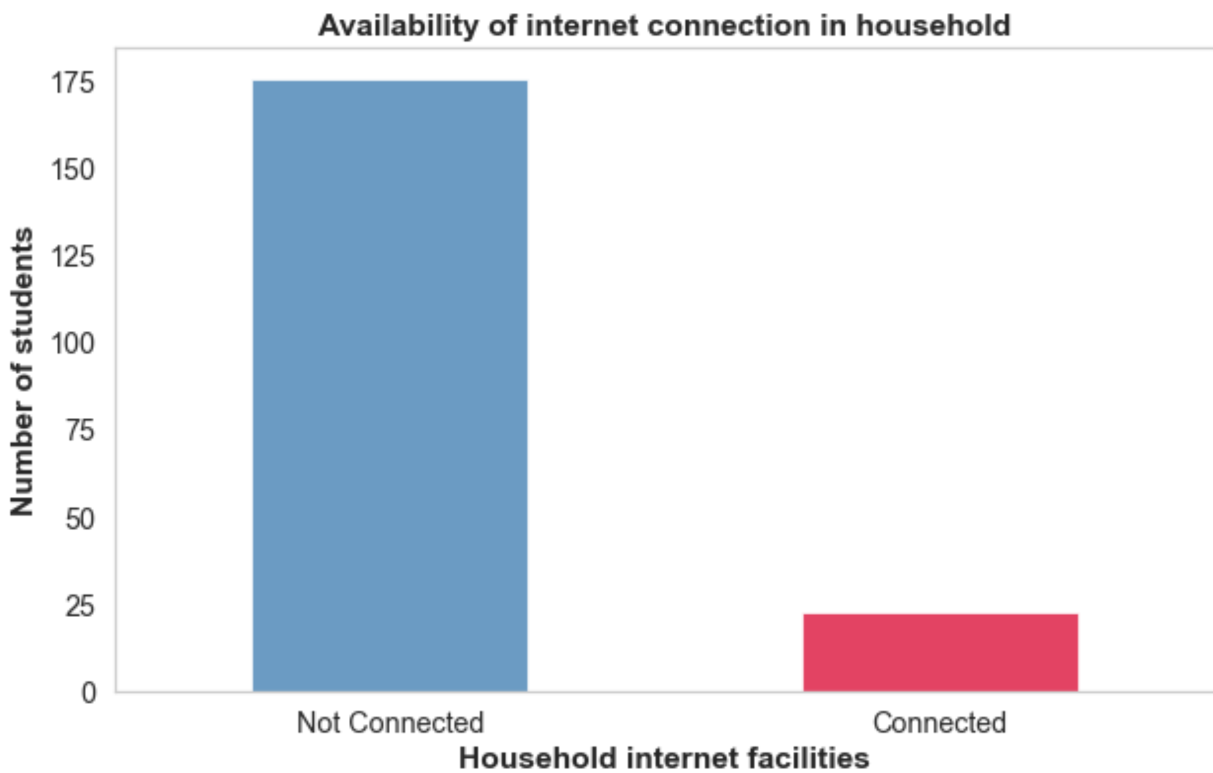
Let's check the histogram.

In [77]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```



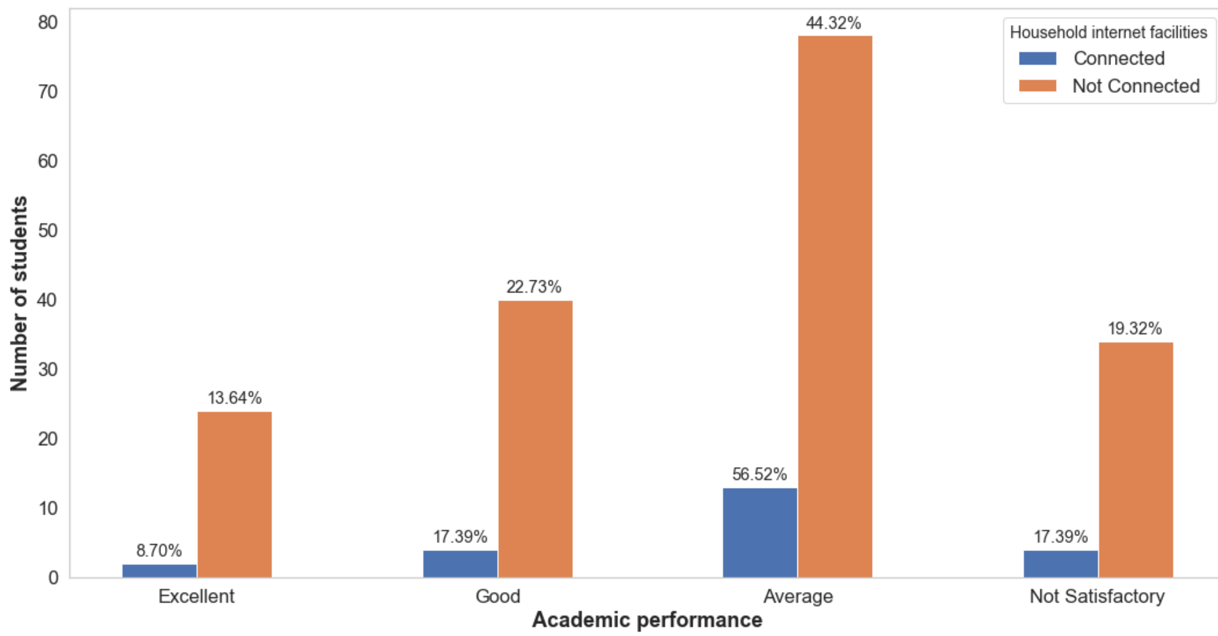
Plotting 'Household Internet Facilities'

In [78]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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'` .

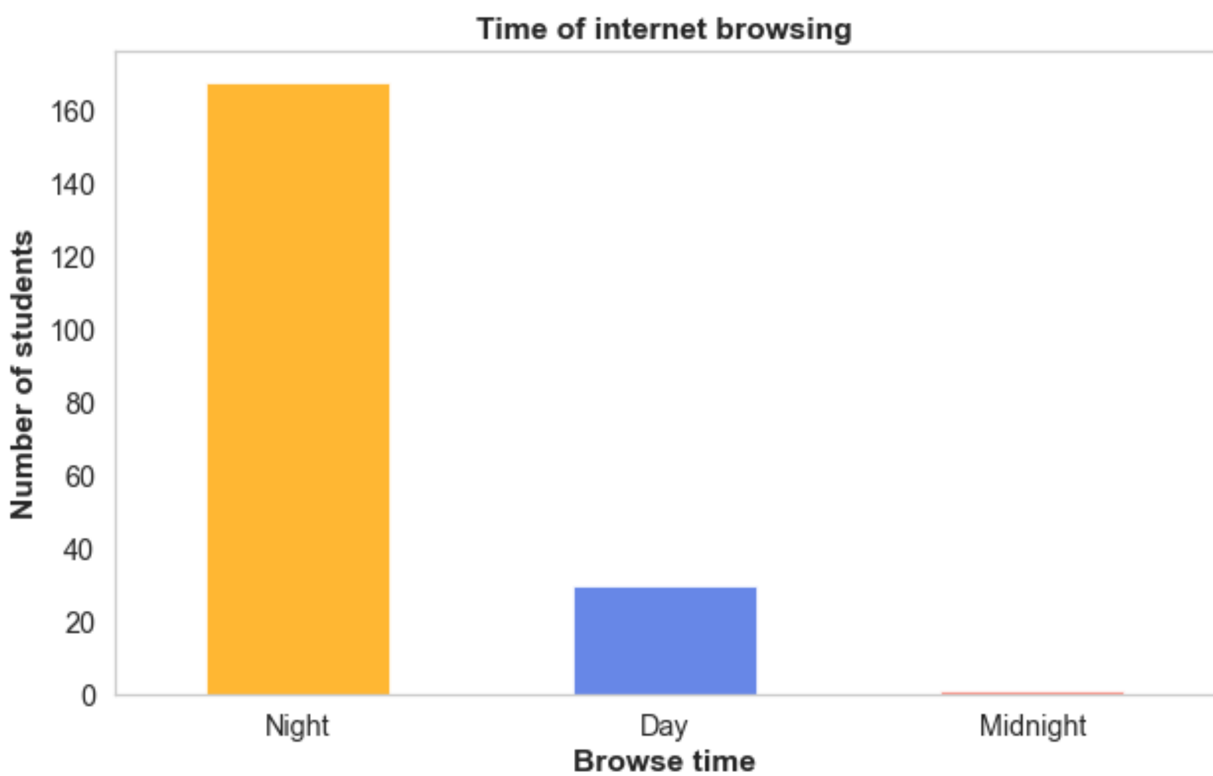
In [79]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Household Internet Facilities',
                                  college_df['Household Internet Facilities'].valu

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()
```



Plotting 'Time Of Internet Browsing'

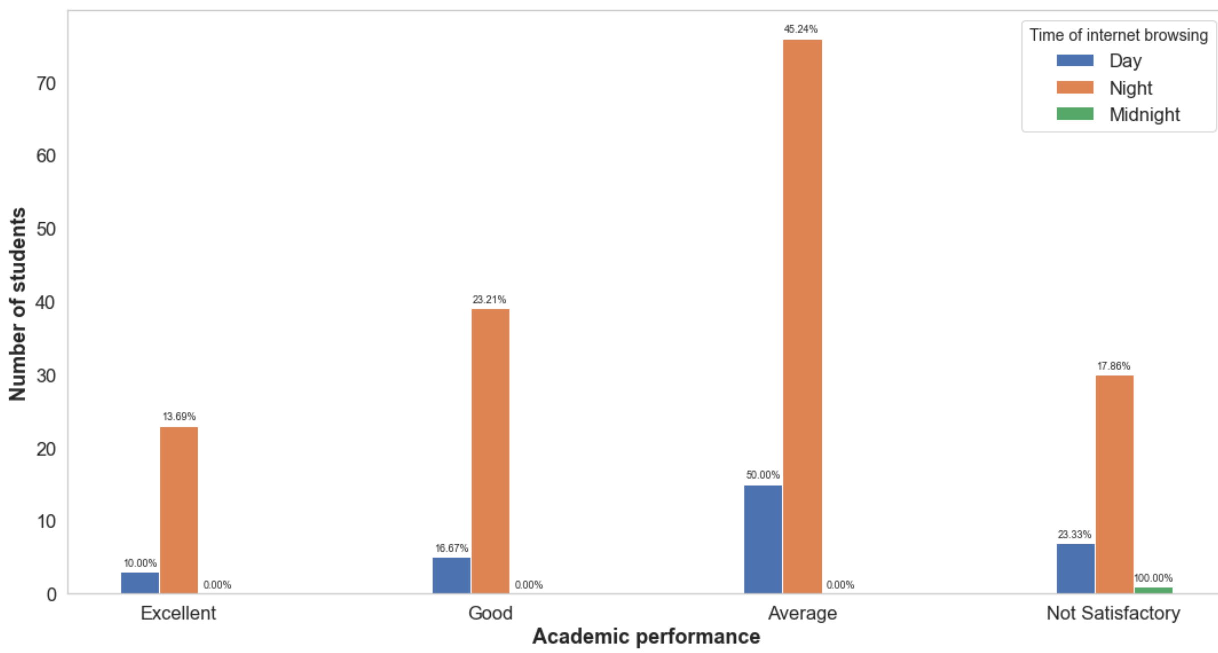
Let's check the histogram.

In [80]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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'` .

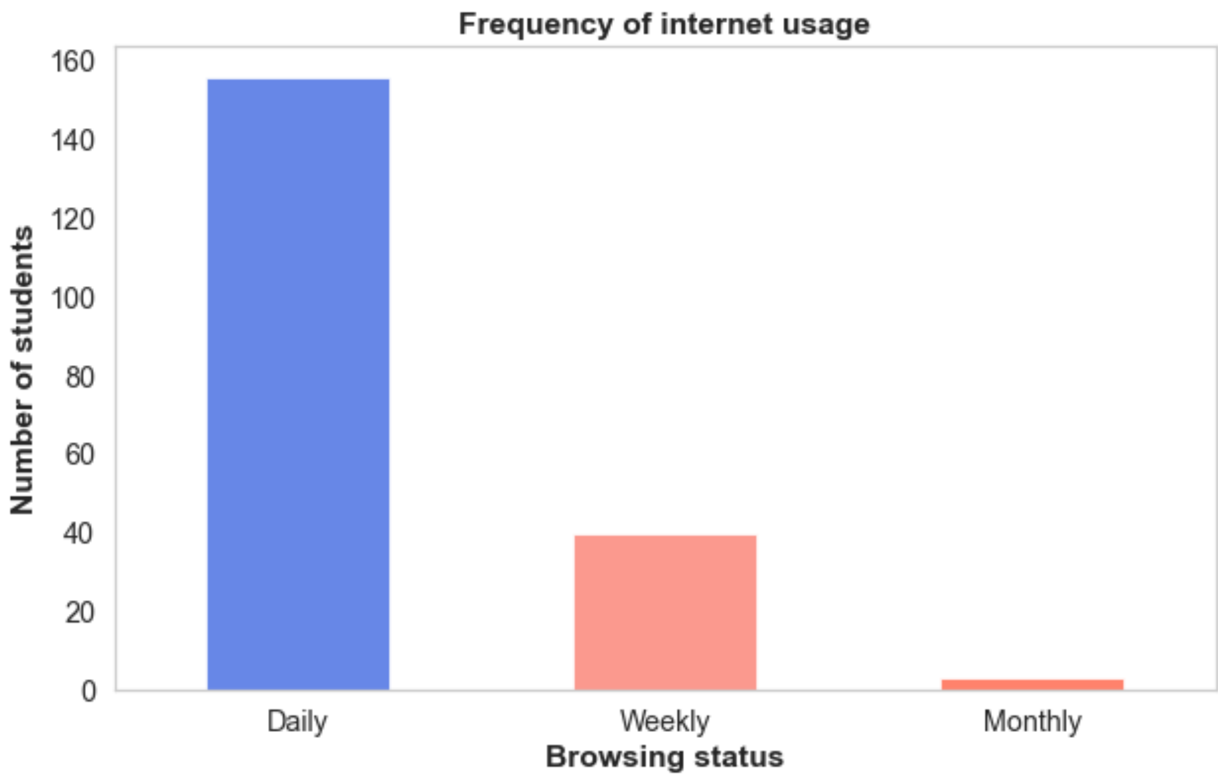
In [81]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Time Of Internet Browsing',
                              ['Day', 'Night', 'Midnight'])

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['Day'], width/2, label = 'Day')
rects2 = ax.bar(x, dictionary['Night'], width/2, label = 'Night')
rects3 = 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)
ax.set_xticklabels(labels)
ax.legend(title='Time of internet browsing', title_fontsize=14)

sns.set(font_scale=0.75)

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

fig.tight_layout()

plt.show()
```



Plotting `'Frequency Of Internet Usage'`

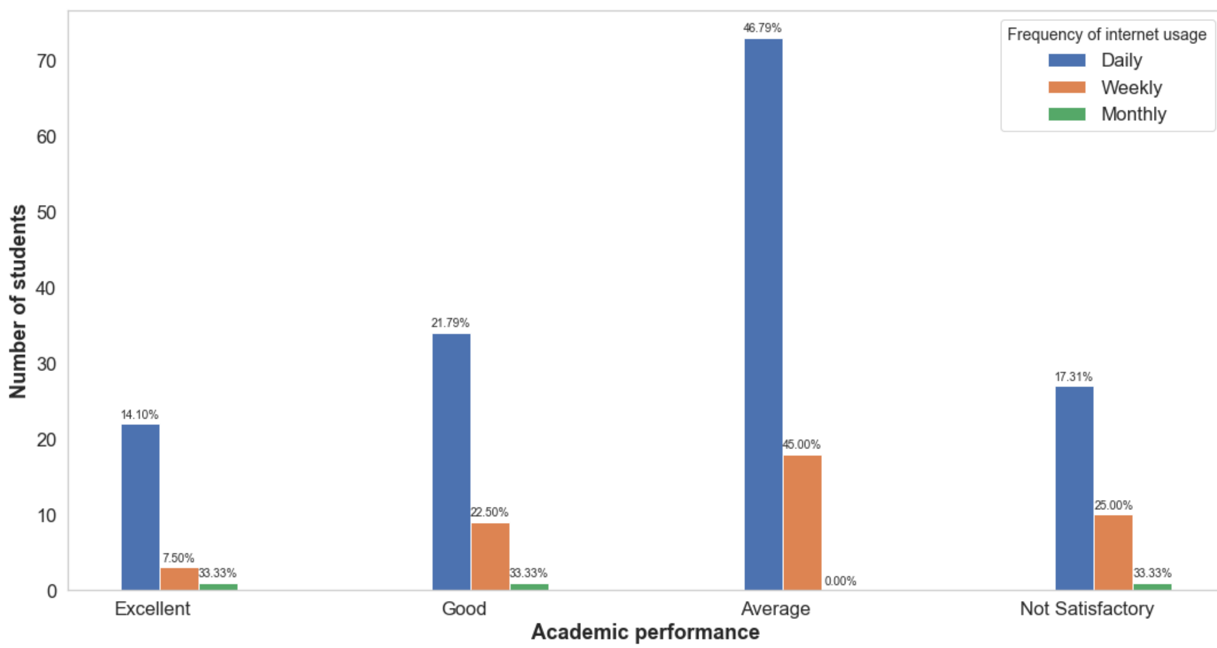
Let's check the histogram.

In [82]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_df['Frequency Of Internet Usage'], color=['royalk
                     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 [83]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_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()
```



Plotting 'Place Of Student's Residence'

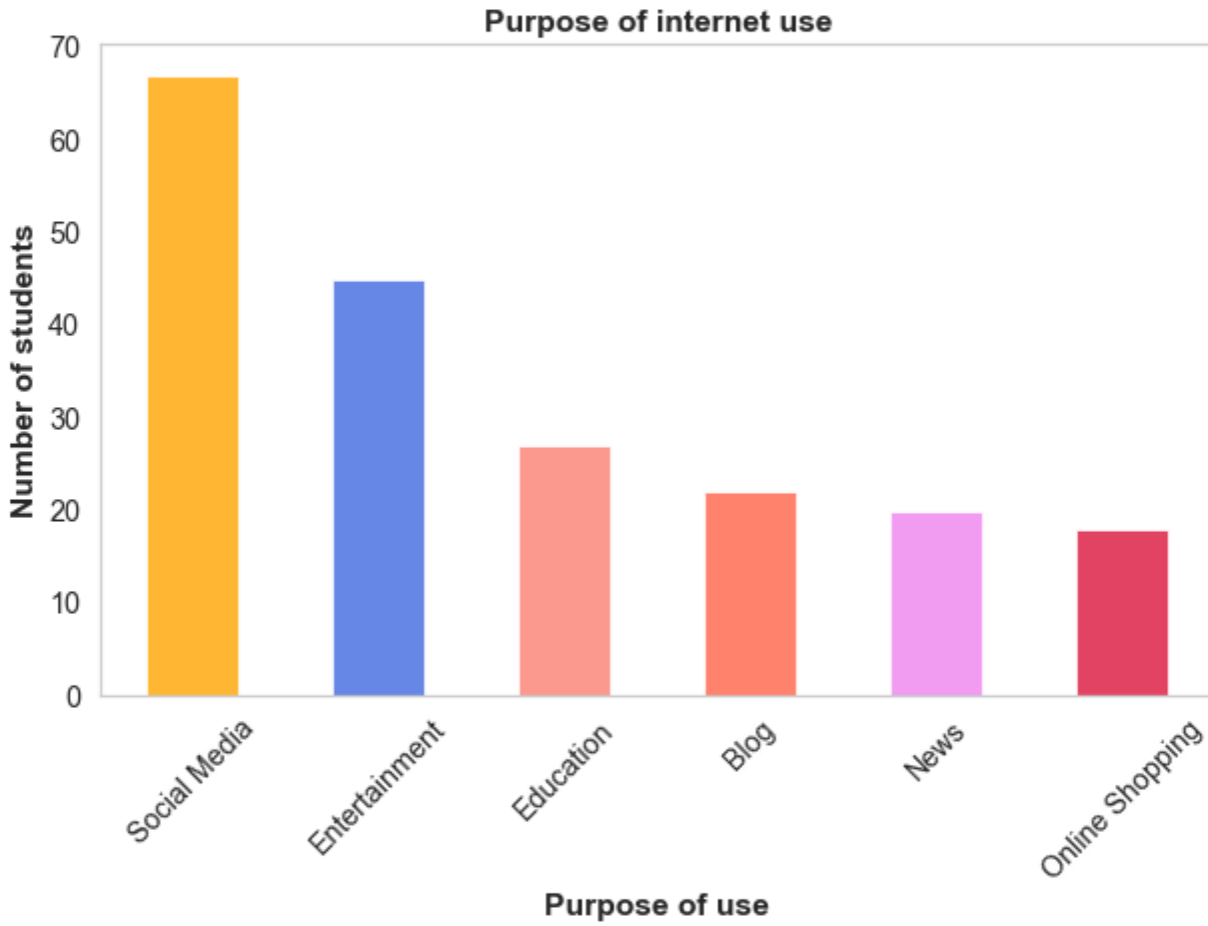
Let's check the histogram.

In [84]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```

Plotting 'Purpose Of Internet Use'

Let's check the histogram.

In [85]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_df['Purpose Of Internet Use'], rot=45,
                     color = ['orange', 'royalblue', 'salmon', 'tomato', 'vio
                     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'` .

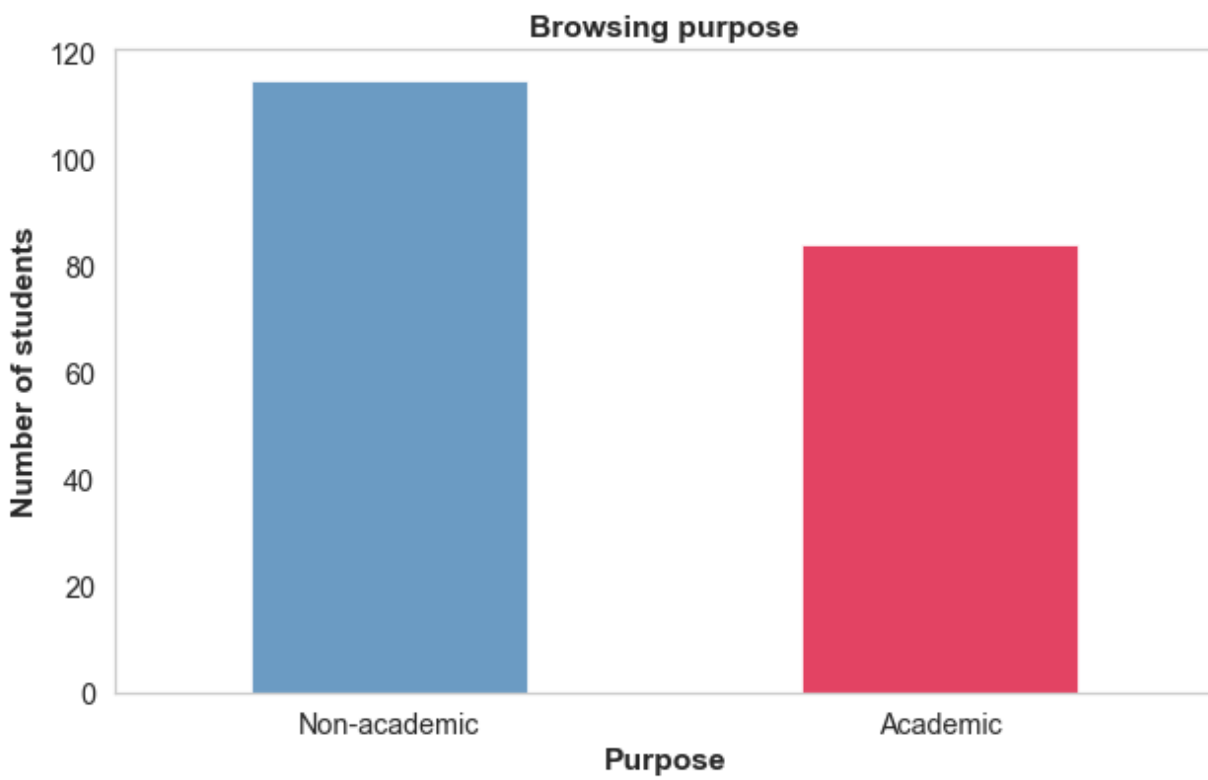
In [86]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Purpose Of Internet Use',
                              college_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 = 'Educatic
rects3 = ax.bar(x - width/2, dictionary['Entertainment'], width/2, label = 'Er
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 W.R.T. Academic Performance', fontwei
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Purpose of internet use', title_fontsize=16)

sns.set(font_scale=0.75)

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

fig.tight_layout()

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

```
Saving figure Purpose_Of_Internet_Use_WRT_Academic_Performance_Histogram
```

Plotting 'Browsing Purpose'

Let's check the histogram.

In [87]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```

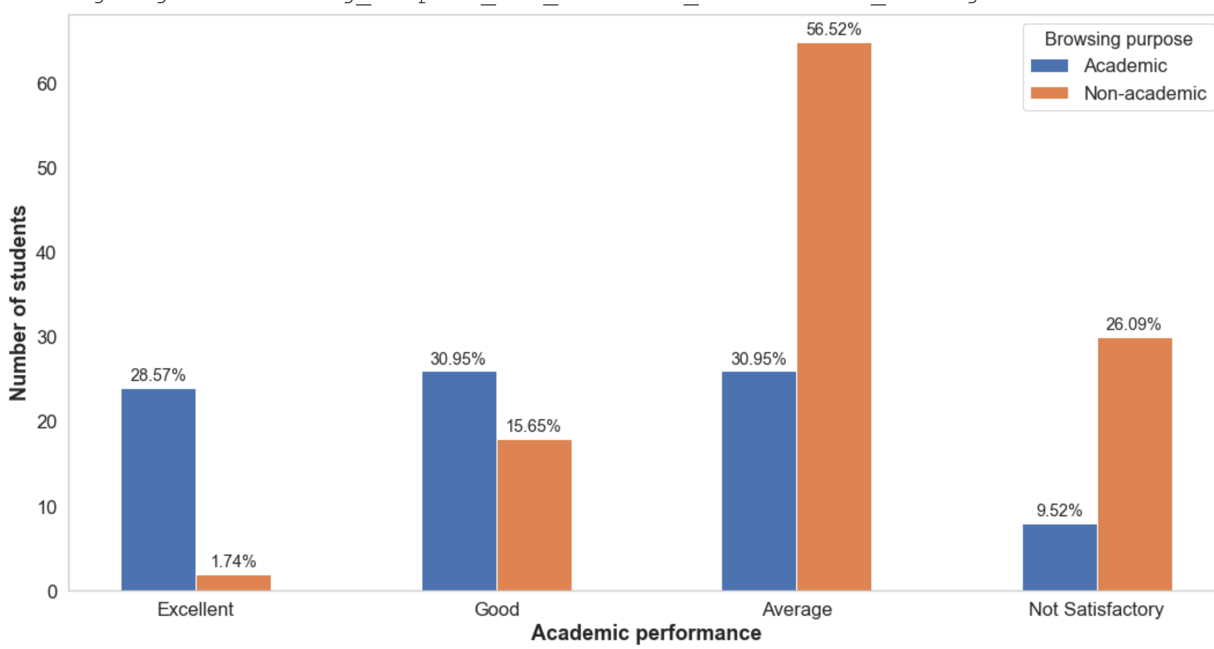


Let's check the distribution of this feature against the target i.e. `'Academic Performance'` .

In [88]:

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

dictionary = cat_vs_cat_bar_plot(college_df, 'Browsing Purpose',
                                 college_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 W.R.T. Academic Performance', fontweight = '
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_Histogram')
plt.show()
```

```
Saving figure Browsing_Purpose_WRT_Academic_Performance_Histogram
```



Plotting 'Webinar'

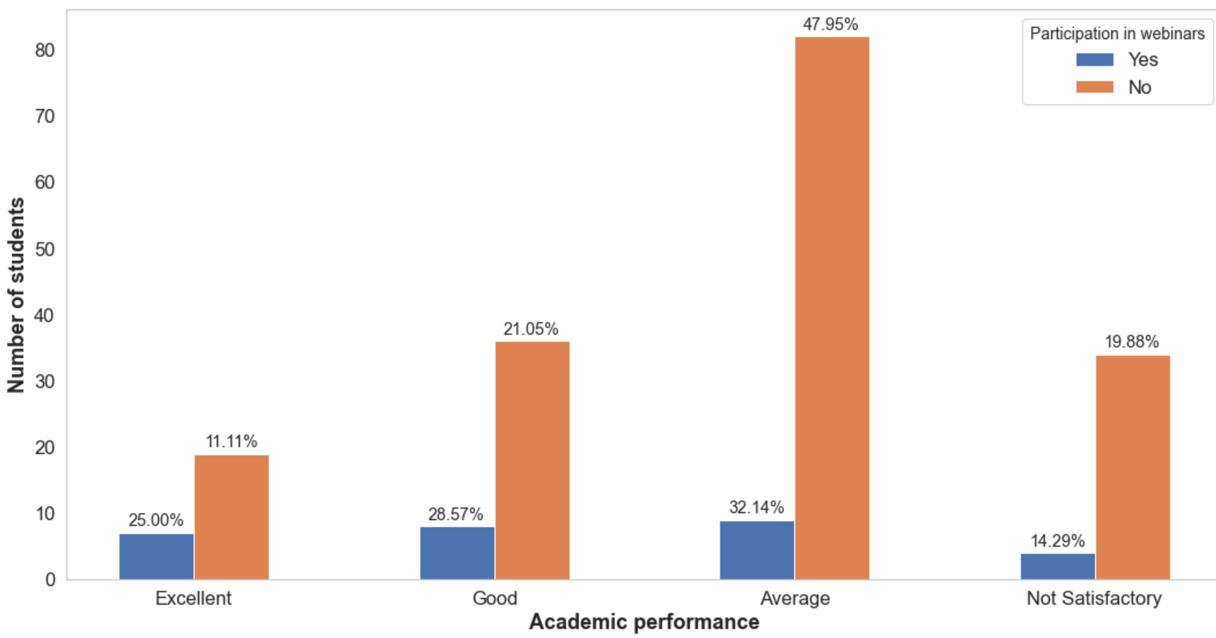
Let's check the histogram.

In [89]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_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 [90]:

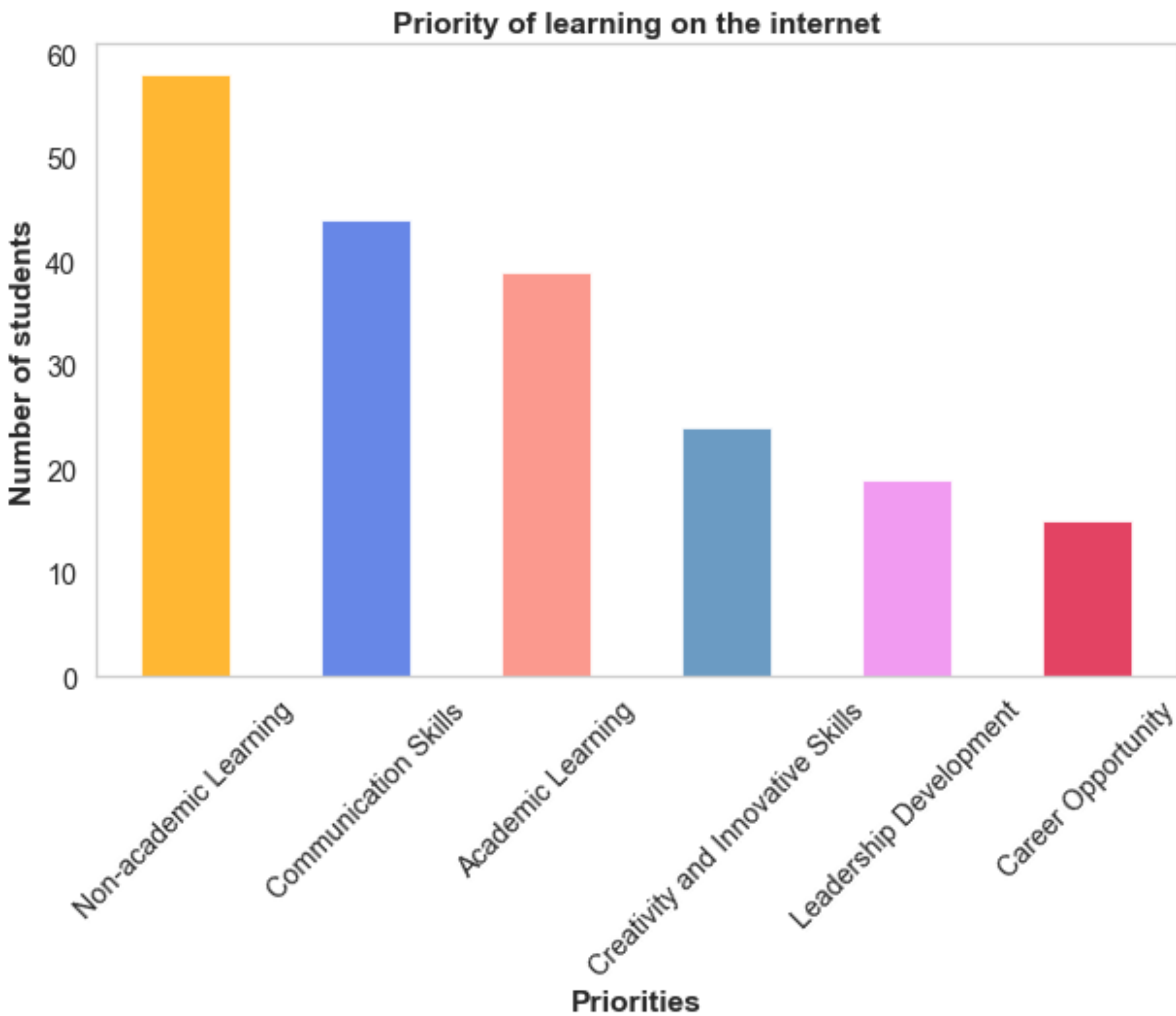
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Webinar',
                                 college_df['Webinar'].value_counts().index.tolis

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()
```



Plotting 'Priority Of Learning On The Internet'

Let's check the histogram.

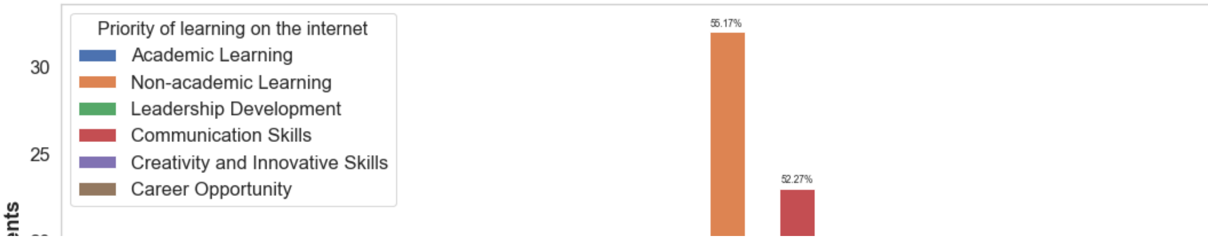
In [91]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```



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

In [92]:
```
sns.set(font_scale=1.5)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Priority Of Learning On The Inte
                              ['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 W.R.T. Academic Performa
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=0.7)

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

fig.tight_layout()

save_fig('Priority_Of_Learning_On_The_Internet_WRT_Academic_Performance_Histog

plt.show()
```

```
Saving figure Priority_Of_Learning_On_The_Internet_WRT_Academic_Performance_Hi
stogram
```

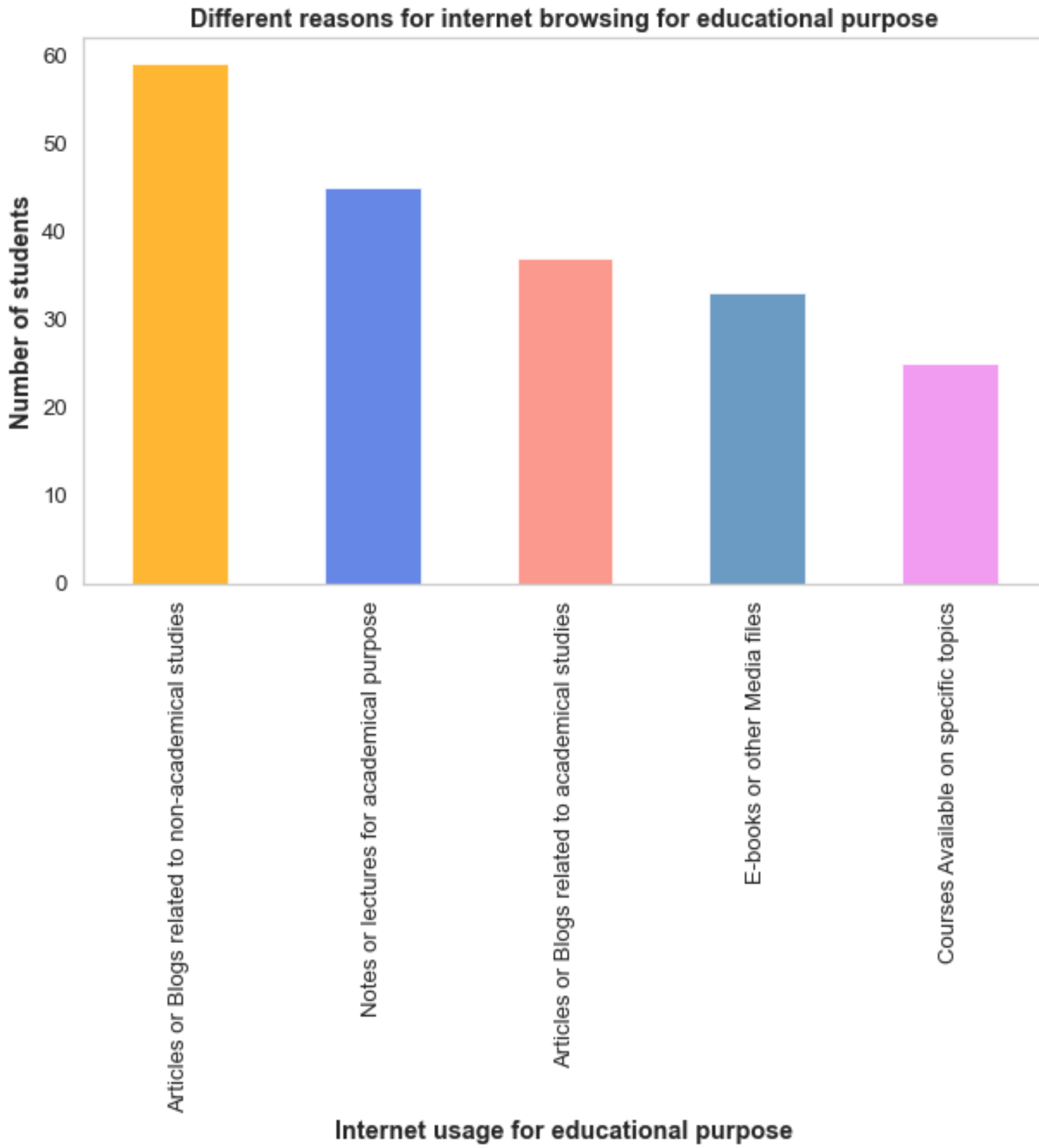


Plotting 'Internet Usage For Educational Purpose'

Let's check the histogram.

In [93]:

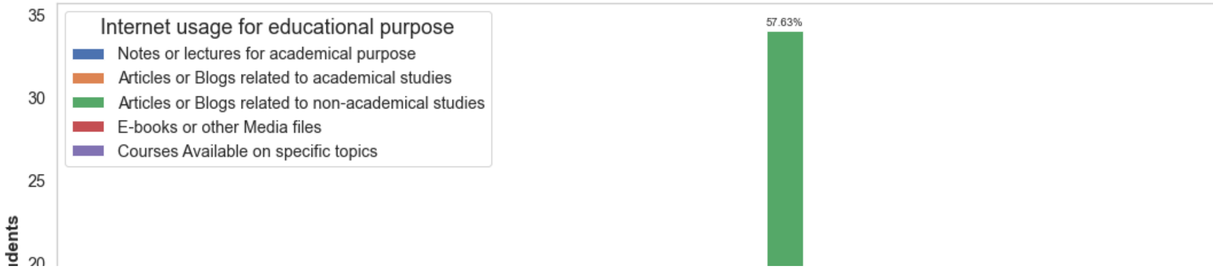
```
plt.figure(figsize=(10, 11))
plt.subplots_adjust(top=0.5, bottom=0.1, hspace=0.5, wspace=0.2)
sns.set(font_scale=1.2)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```



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

In [94]:
```
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot(college_df, 'Internet Usage For Educational I
                              college_df['Internet Usage For Educational Purpo

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 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 performance', fontweight = 'bold')
# ax.set_title('Internet Usage For Educational Purpose W.R.T. Academic Perforn
ax.set_xticks(x - width/2)
ax.set_xticklabels(labels)
ax.legend(title='Internet usage for educational purpose', title_fontsize=18, 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_Hist

plt.show()
```

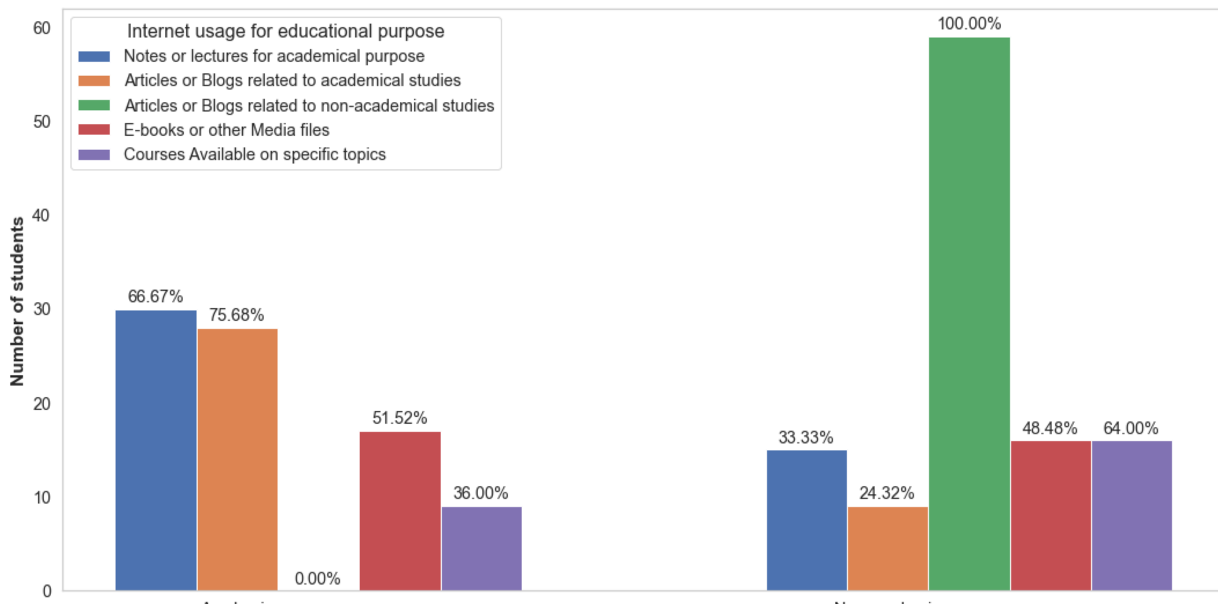
```
Saving figure Internet_Usage_For_Educational_Purpose_WRT_Academic_Performance_
Histogram
```



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

In [95]:
```
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

dictionary = cat_vs_cat_bar_plot_browsing_purpose(college_df, 'Internet Usage
                              college_df['Internet Usage For Educational Purpo

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 stuc
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('Browsing purpose', fontweight = 'bold')
# ax.set_title('Internet Usage For Educational Purpose W.R.T. Browsing Purpose
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_Histogra

plt.show()
```

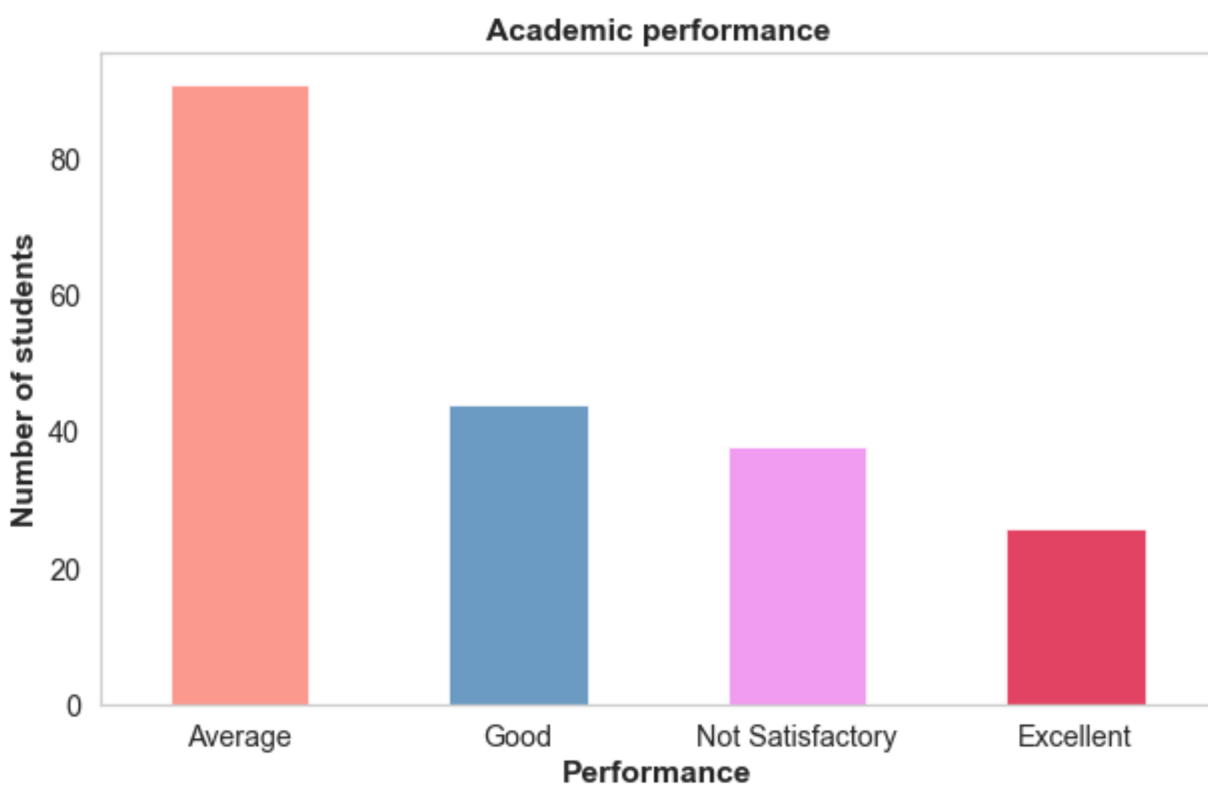
```
Saving figure Internet_Usage_For_Educational_Purpose_WRT_Browsing_Purpose_Hist
ogram
```



Plotting 'Academic Performance'

Let's check the histogram.

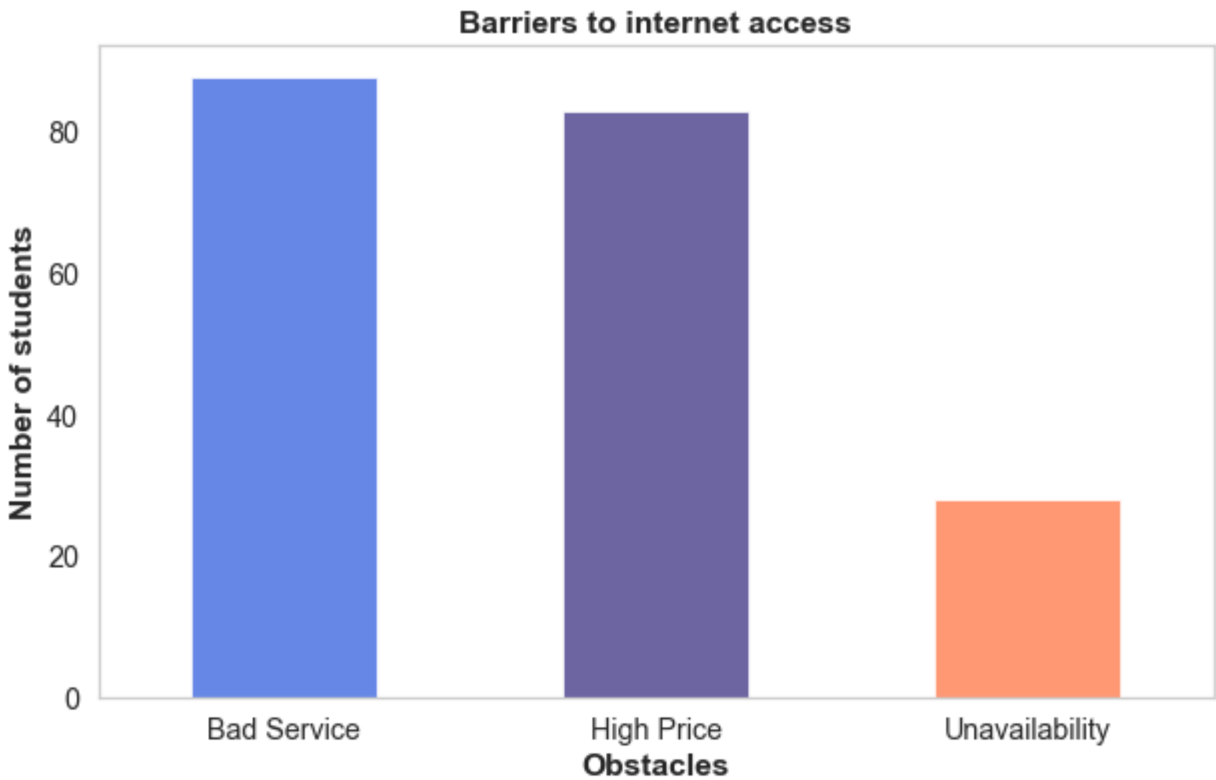
In [96]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

categorical_bar_plot(college_df['Academic Performance'], color=['salmon', 'ste
                     title='Academic performance', xlabel='Performance')

plt.show()
```



Plotting 'Barriers To Internet Access'

Let's check the histogram.

In [97]:
```
plt.figure(figsize=(10, 6))
sns.set(font_scale=1.3)
sns.set_style("whitegrid", {'axes.grid' : False})

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

plt.show()
```



Inspecting Age Closer

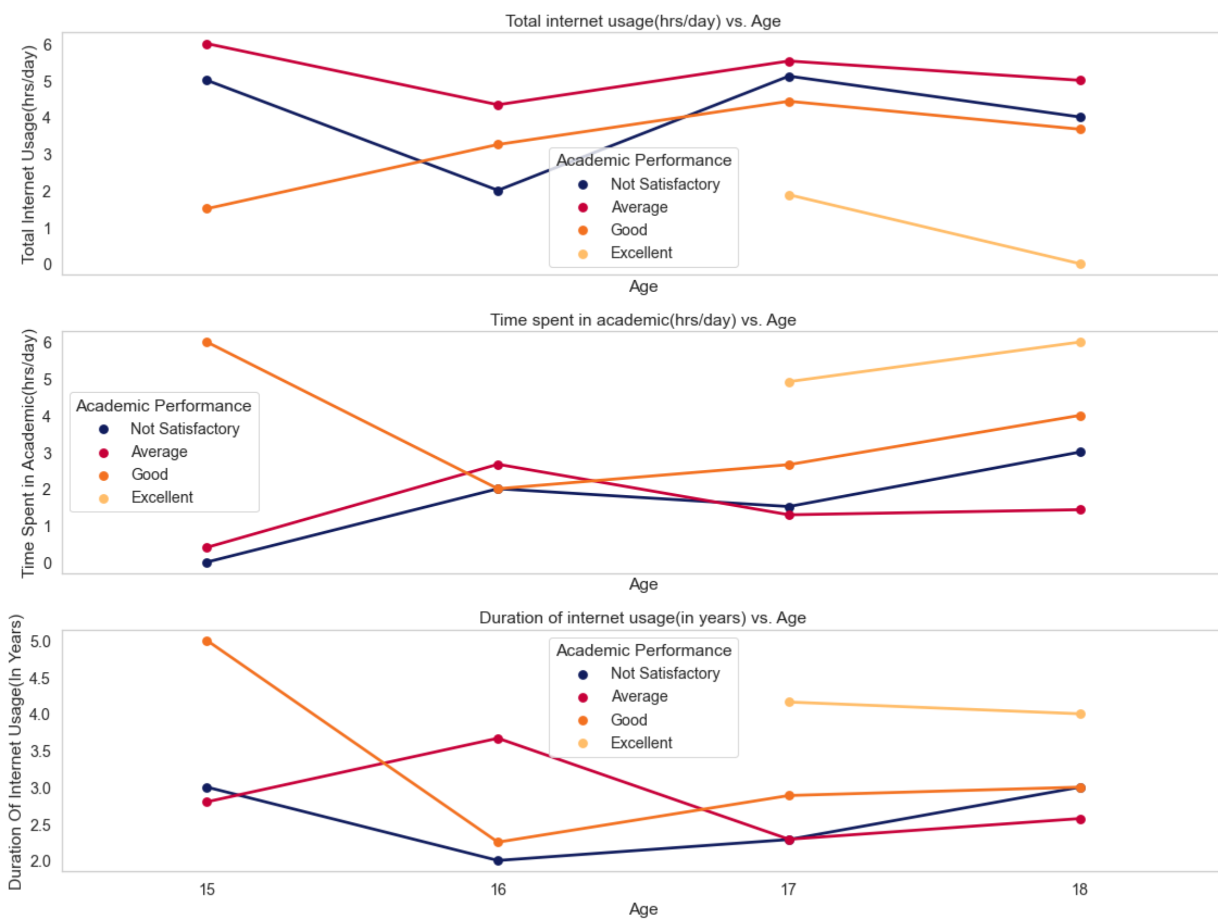
Let's define a function to make this process easier.

In [98]:
```
# For Styling:
cust_palt = [
    '#111d5e', '#c70039', '#f37121', '#ffbd69', '#ffc93c'
]
```

In [99]:
```
def ctn_freq(dataframe, cols, xax, hue = None, rows = 3, columns = 1):

    ''' 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 [100…
```
ctn_freq(college_df,
         ['Total Internet Usage(hrs/day)', 'Time Spent in Academic(hrs/day)',
         'Age', hue='Academic Performance',rows=3, columns=1)
```

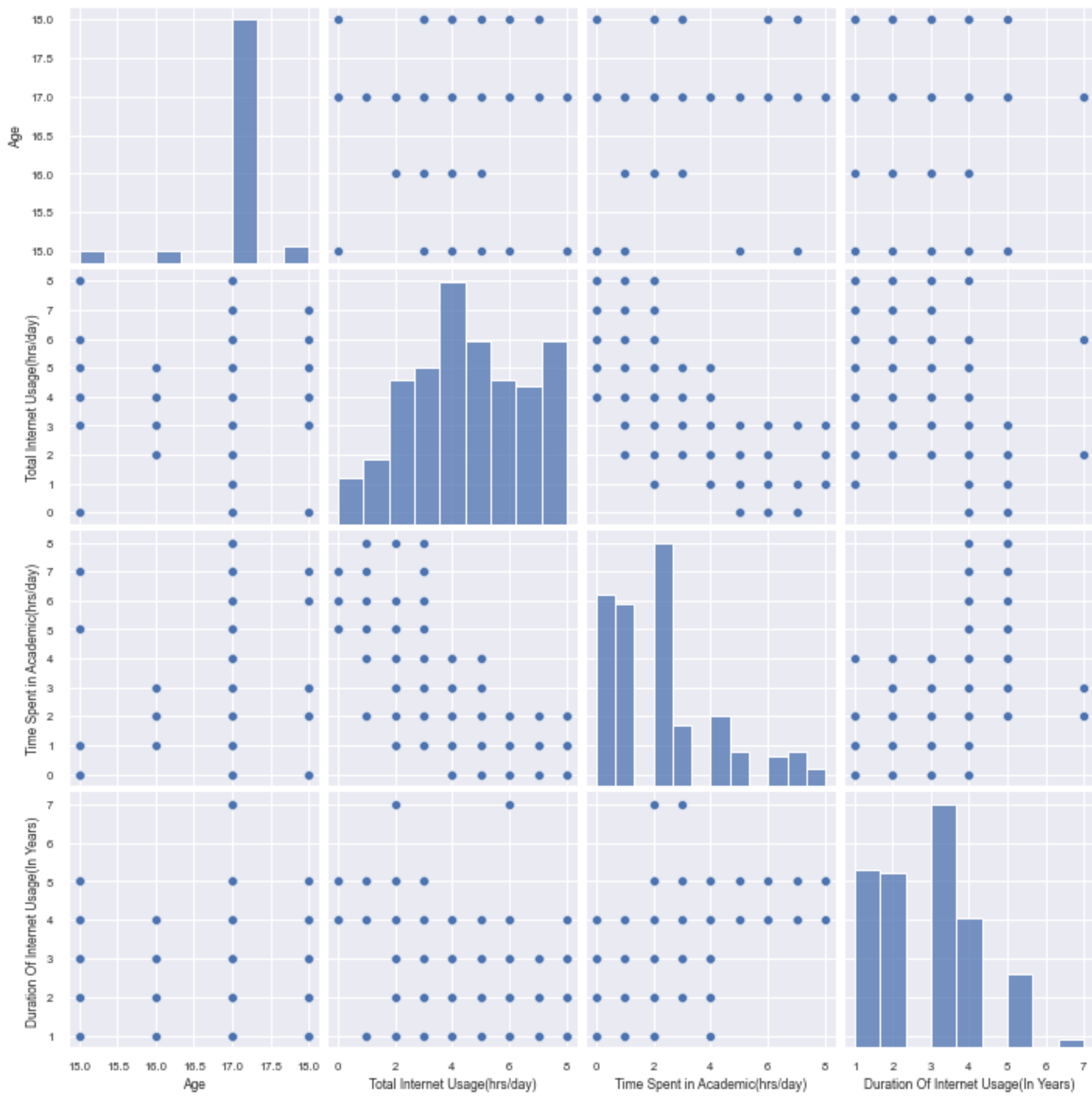


Multivariate Analysis

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 [101…

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

sns.set(font_scale = 0.7)
sns.pairplot(college_df)

plt.show()
```



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

In [102…

```
sns.set(font_scale = 0.7)
sns.pairplot(college_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 [103…

```
college_df.corr(method='pearson', min_periods=1)
```

Out[103…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)
Age	1.000000	0.005744	0.053339	-0.070581
Total Internet Usage(hrs/day)	0.005744	1.000000	-0.652445	-0.465673
Time Spent in Academic(hrs/day)	0.053339	-0.652445	1.000000	0.562748

	Age	Total Internet	Time Spent in	Duration Of Internet Usage(In

In [104…

```
# Correlation heatmap between variables:

sns.set(font_scale=1.5)

correlation_df = college_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 [105…

```
college_labels = college_df["Academic Performance"].copy()

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

college_df.head()
```

Out[105…

	Gender	Age	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	I St Re
0	Female	17	Google	Very Effective	Mobile	Home	Not Connected	Night	Daily	
1	Female	17	Facebook	Effective	Mobile	Home	Not Connected	Night	Daily	
2	Female	17	Youtube	Very Effective	Mobile	Home	Not Connected	Night	Daily	
3	Female	18	Youtube	Effective	Mobile	Home	Not Connected	Night	Weekly	
4	Male	17	Whatsapp	Very Effective	Mobile	Home	Not Connected	Night	Daily	

In [106…

```
college_labels.head()
```

Out[106…

```
0    Not Satisfactory
1             Average
2             Average
3             Average
4             Average
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 [107…

```
college_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 19 columns):
 #   Column                              Non-Null Count  Dtype 
---  ------                              --------------  ----- 
 0   Gender                              199 non-null    object
 1   Age                                 199 non-null    int64 
```

```
 2   Frequently Visited Website                199 non-null    object
 3   Effectiveness Of Internet Usage           199 non-null    object
 4   Devices Used For Internet Browsing        199 non-null    object
 5   Location Of Internet Use                  199 non-null    object
 6   Household Internet Facilities             199 non-null    object
 7   Time Of Internet Browsing                 199 non-null    object
 8   Frequency Of Internet Usage               199 non-null    object
 9   Place Of Student's 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 Internet Use                   199 non-null    object
 13  Duration Of Internet Usage(In Years)      199 non-null    int64
 14  Browsing Purpose                          199 non-null    object
 15  Priority Of Learning On The Internet      199 non-null    object
 16  Webinar                                   199 non-null    object
 17  Internet Usage For Educational Purpose    199 non-null    object
 18  Barriers To Internet Access               199 non-null    object
dtypes: int64(4), object(15)
memory usage: 29.7+ 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 [108…

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

college_cat.head()
```

Out[108…

	Gender	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 (Student Residen
0	Female	Google	Very Effective	Mobile	Home	Not Connected	Night	Daily	Tow
1	Female	Facebook	Effective	Mobile	Home	Not Connected	Night	Daily	Tow
2	Female	Youtube	Very Effective	Mobile	Home	Not Connected	Night	Daily	Tow
3	Female	Youtube	Effective	Mobile	Home	Not Connected	Night	Weekly	Tow
4	Male	Whatsapp	Very Effective	Mobile	Home	Not Connected	Night	Daily	Tow

In [109…
```
college_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 15 columns):
 #   Column                                 Non-Null Count  Dtype 
---  ------                                 --------------  ----- 
 0   Gender                                 199 non-null    object
 1   Frequently Visited Website             199 non-null    object
 2   Effectiveness Of Internet Usage        199 non-null    object
 3   Devices Used For Internet Browsing     199 non-null    object
 4   Location Of Internet Use               199 non-null    object
 5   Household Internet Facilities          199 non-null    object
 6   Time Of Internet Browsing              199 non-null    object
 7   Frequency Of Internet Usage            199 non-null    object
 8   Place Of Student's Residence           199 non-null    object
 9   Purpose Of Internet Use                199 non-null    object
 10  Browsing Purpose                       199 non-null    object
 11  Priority Of Learning On The Internet   199 non-null    object
 12  Webinar                                199 non-null    object
 13  Internet Usage For Educational Purpose 199 non-null    object
 14  Barriers To Internet Access            199 non-null    object
dtypes: object(15)
memory usage: 23.4+ KB
```

Store the numerical attributes in a separate variable.

In [110…
```
college_num = college_df[['Age', 'Total Internet Usage(hrs/day)','Time Spent i
                          'Duration Of Internet Usage(In Years)']].copy()

college_num.head()
```

Out[110…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)
0	17	2	4	3
1	17	8	0	3
2	17	6	2	3
3	18	7	0	2
4	17	4	4	4

In [111…
```
college_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 199 entries, 0 to 198
Data columns (total 4 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Age                                   199 non-null    int64
 1   Total Internet Usage(hrs/day)         199 non-null    int64
 2   Time Spent in Academic(hrs/day)       199 non-null    int64
 3   Duration Of Internet Usage(In Years)  199 non-null    int64
dtypes: int64(4)
memory usage: 6.3 KB
```

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

In [112…
```
from sklearn import preprocessing

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

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

temp_df_cat.head()
```

Out[112…

	Gender	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 (Student Residen
0	0	2	2	2	2	1	2	0	
1	0	0	0	2	2	1	2	0	
2	0	5	2	2	2	1	2	0	
3	0	5	0	2	2	1	2	2	
4	1	4	2	2	2	1	2	0	

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

In [113…
```
# 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 [114…
```
# Place the DataFrames side by side

X = pd.concat([college_num, temp_df_cat], axis=1)
y = college_labels

X.head()
```

Out[114…

	Age	Total Internet Usage(hrs/day)	Time Spent in Academic(hrs/day)	Duration Of Internet Usage(In Years)	Gender	Frequently Visited Website	Effectiveness Of Internet Usage	Devices Used For Internet Browsing
0	17	2	4	3	0	2	2	2
1	17	8	0	3	0	0	0	2
2	17	6	2	3	0	5	2	2
3	18	7	0	2	0	5	0	2

Duration Devices

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

In [115…
```
# 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)
```

```
(139, 19) (60, 19) (139,) (60,)
```

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 [116…
```
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:  0.5179856115107914
```

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

In [117…
```
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: 0.35
```

```
[[13  3  0 12]
 [ 0  5  0  1]
 [ 6  7  1  2]
 [ 5  3  0  2]]
                  precision    recall  f1-score   support

         Average       0.54      0.46      0.50        28
       Excellent       0.28      0.83      0.42         6
            Good       1.00      0.06      0.12        16
Not Satisfactory       0.12      0.20      0.15        10

        accuracy                           0.35        60
       macro avg       0.48      0.39      0.30        60
    weighted avg       0.57      0.35      0.33        60
```

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

In [118…

```python
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_sgd = KFold(n_splits=10, shuffle=True, random_state=42)
cross_val_score(sgd_clf, X_train, y_train, cv=cv_sgd, scoring="accuracy", n_jo
```

Out[118…

```
array([0.64285714, 0.64285714, 0.35714286, 0.5       , 0.42857143,
       0.57142857, 0.28571429, 0.64285714, 0.5       , 0.61538462])
```

In [119…

```python
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: 0.333 (0.224)
```

Let's check the score.

In [120…

```python
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: 0.317 (0.029)
```

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 [121…

```
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 "
```

Let's check the `scores` variable.

In [122…

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

```
0 : [0.4676258992805755, 0.4666666666666667]
1 : [0.4172661870503597, 0.2833333333333333]
2 : [0.5827338129496403, 0.26666666666666666]
3 : [0.5107913669064749, 0.3]
```

```
4 : [0.5539568345323741, 0.43333333333333335]
5 : [0.49640287769784175, 0.26666666666666666]
6 : [0.5179856115107914, 0.45]
7 : [0.45323741007194246, 0.23333333333333334]
8 : [0.5179856115107914, 0.35]
9 : [0.5179856115107914, 0.35]
10 : [0.5179856115107914, 0.35]
11 : [0.5179856115107914, 0.35]
12 : [0.5179856115107914, 0.35]
13 : [0.5179856115107914, 0.35]
14 : [0.5179856115107914, 0.35]
15 : [0.5179856115107914, 0.35]
16 : [0.5179856115107914, 0.35]
17 : [0.5179856115107914, 0.35]
18 : [0.5179856115107914, 0.35]
19 : [0.5179856115107914, 0.35]
20 : [0.5179856115107914, 0.35]
21 : [0.5179856115107914, 0.35]
```

Finally, let's plot the training score.

In [123…

```
# 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 [124…

```
# 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 [125…

```
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 [126…

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

dec_tree_clf = DecisionTreeClassifier(max_depth=12, max_leaf_nodes = 50, rando

dec_tree_clf.fit(X_train, y_train)

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

```
Training score:  0.9640287769784173
```

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

In [127…

```
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: 0.36666666666666664
[[15  1  6  6]
 [ 2  2  2  0]
 [ 7  4  3  2]
 [ 6  0  2  2]]
                  precision    recall  f1-score   support

         Average       0.50      0.54      0.52        28
       Excellent       0.29      0.33      0.31         6
            Good       0.23      0.19      0.21        16
Not Satisfactory       0.20      0.20      0.20        10

        accuracy                           0.37        60
       macro avg       0.30      0.31      0.31        60
```

```
weighted avg       0.36      0.37      0.36        60
```

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

In [128…

```python
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[128…

```
array([0.14285714, 0.42857143, 0.28571429, 0.42857143, 0.28571429,
       0.28571429, 0.28571429, 0.14285714, 0.21428571, 0.46153846])
```

In [129…

```python
scores = cross_val_score(dec_tree_clf, X_test, y_test, cv=cv_dec_tree, scoring
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.400 (0.170)
```

Let's check the score.

In [130…

```python
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: 0.400 (0.071)
```

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 [131…

```python
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 [132…

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

```
0 : [0.5467625899280576, 0.5]
1 : [0.5755395683453237, 0.5]
2 : [0.60431654676259, 0.38333333333333336]
3 : [0.6546762589928058, 0.4666666666666667]
4 : [0.697841726618705, 0.45]
5 : [0.7697841726618705, 0.45]
6 : [0.8057553956834532, 0.4]
```

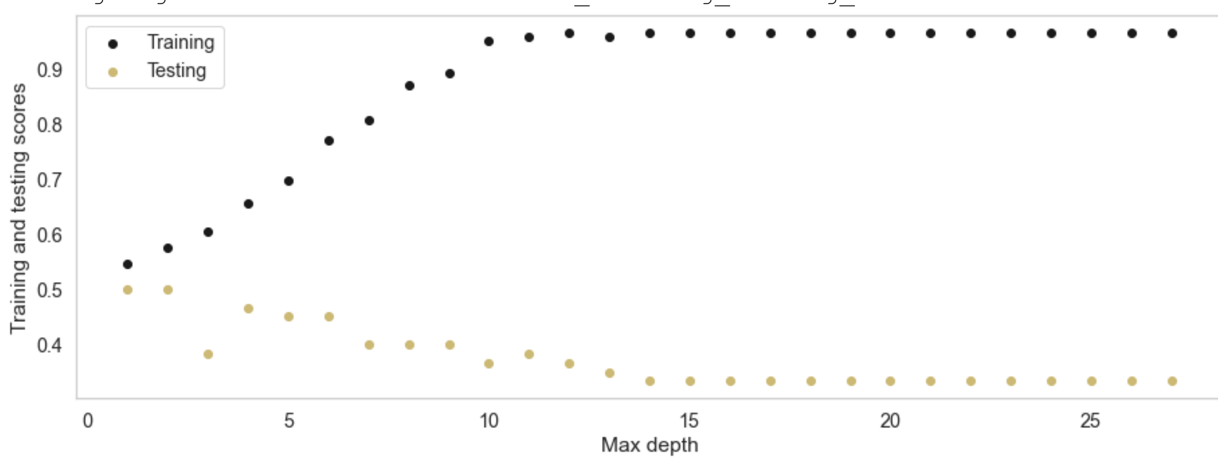
```
7 : [0.8705035971223022, 0.4]
8 : [0.8920863309352518, 0.4]
9 : [0.9496402877697842, 0.36666666666666664]
10 : [0.9568345323741008, 0.38333333333333336]
11 : [0.9640287769784173, 0.36666666666666664]
12 : [0.9568345323741008, 0.35]
13 : [0.9640287769784173, 0.3333333333333333]
14 : [0.9640287769784173, 0.3333333333333333]
15 : [0.9640287769784173, 0.3333333333333333]
16 : [0.9640287769784173, 0.3333333333333333]
17 : [0.9640287769784173, 0.3333333333333333]
18 : [0.9640287769784173, 0.3333333333333333]
19 : [0.9640287769784173, 0.3333333333333333]
20 : [0.9640287769784173, 0.3333333333333333]
21 : [0.9640287769784173, 0.3333333333333333]
22 : [0.9640287769784173, 0.3333333333333333]
23 : [0.9640287769784173, 0.3333333333333333]
24 : [0.9640287769784173, 0.3333333333333333]
25 : [0.9640287769784173, 0.3333333333333333]
26 : [0.9640287769784173, 0.3333333333333333]
```

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

In [133…

```
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 [134…

```
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:  0.6330935251798561
```

```
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(
```

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

In [135…

```
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: 0.45
[[19  2  4  3]
 [ 1  2  3  0]
 [ 8  3  5  0]
 [ 7  0  2  1]]
                  precision    recall  f1-score   support

         Average       0.54      0.68      0.60        28
       Excellent       0.29      0.33      0.31         6
            Good       0.36      0.31      0.33        16
Not Satisfactory       0.25      0.10      0.14        10

        accuracy                           0.45        60
       macro avg       0.36      0.36      0.35        60
    weighted avg       0.42      0.45      0.42        60
```

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

In [136…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_log_reg = KFold(n_splits=5, shuffle=True, random_state=42)
cross_val_score(log_reg, X_train, y_train, cv=cv_log_reg, scoring="accuracy",
```

Out[136…
```
array([0.5       , 0.5       , 0.28571429, 0.42857143, 0.51851852])
```

In [137…
```
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: 0.467 (0.145)
```

Let's check the score.

In [138…
```
scores = cross_val_score(log_reg, X_test, y_test, cv=3, scoring="accuracy", n_
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.350 (0.108)
```

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 [139…
```
m_iter = []
training = []
test = []
scores = {}

max_i = [50, 70, 90, 100, 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
  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 [140…

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

```
0 : [0.6115107913669064, 0.45]
1 : [0.6330935251798561, 0.45]
2 : [0.6330935251798561, 0.43333333333333335]
3 : [0.6330935251798561, 0.43333333333333335]
```

```
4 : [0.6402877697841727, 0.43333333333333335]
5 : [0.6258992805755396, 0.45]
6 : [0.6330935251798561, 0.45]
7 : [0.6330935251798561, 0.45]
8 : [0.6330935251798561, 0.45]
9 : [0.6330935251798561, 0.43333333333333335]
10 : [0.6330935251798561, 0.45]
11 : [0.6330935251798561, 0.45]
12 : [0.6258992805755396, 0.43333333333333335]
13 : [0.6258992805755396, 0.43333333333333335]
14 : [0.6258992805755396, 0.43333333333333335]
15 : [0.6258992805755396, 0.43333333333333335]
16 : [0.6258992805755396, 0.43333333333333335]
17 : [0.6258992805755396, 0.45]
18 : [0.6258992805755396, 0.43333333333333335]
19 : [0.6258992805755396, 0.43333333333333335]
20 : [0.6258992805755396, 0.45]
```

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

In [141…

```python
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 [142…
```
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:  1.0
```

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

In [143…
```
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: 0.43333333333333335
[[22  2  2  2]
 [ 2  1  3  0]
 [10  2  3  1]
 [ 8  1  1  0]]
                  precision    recall  f1-score   support

         Average       0.52      0.79      0.63        28
       Excellent       0.17      0.17      0.17         6
            Good       0.33      0.19      0.24        16
Not Satisfactory       0.00      0.00      0.00        10

        accuracy                           0.43        60
       macro avg       0.26      0.28      0.26        60
    weighted avg       0.35      0.43      0.37        60
```

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

In [144…
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_rand_for = KFold(n_splits=10, shuffle=True, random_state=42)
cross_val_score(random_for_clf, X_train, y_train, cv=cv_rand_for, scoring="acc
```

Out[144…
```
array([0.42857143, 0.5       , 0.57142857, 0.5       , 0.35714286,
       0.42857143, 0.5       , 0.57142857, 0.42857143, 0.61538462])
```

In [145…
```
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: 0.433 (0.226)
```

Let's check the score.

In [146…

```
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: 0.367 (0.075)
```

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 [147…

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

for i in range(1, 35):
    clf = RandomForestClassifier(n_estimators=i, max_depth=50, 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)
    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 [148…

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

```
1 : [0.7985611510791367, 0.4166666666666667]
2 : [0.7913669064748201, 0.43333333333333335]
3 : [0.9136690647482014, 0.45]
4 : [0.8776978417266187, 0.4666666666666667]
5 : [0.9280575539568345, 0.4666666666666667]
6 : [0.9424460431654677, 0.4666666666666667]
7 : [0.9712230215827338, 0.48333333333333334]
8 : [0.9640287769784173, 0.4666666666666667]
9 : [1.0, 0.48333333333333334]
10 : [0.9928057553956835, 0.48333333333333334]
11 : [1.0, 0.4666666666666667]
12 : [0.9928057553956835, 0.43333333333333335]
13 : [1.0, 0.43333333333333335]
14 : [1.0, 0.45]
15 : [1.0, 0.43333333333333335]
16 : [1.0, 0.45]
17 : [1.0, 0.4]
18 : [1.0, 0.4166666666666667]
19 : [1.0, 0.43333333333333335]
20 : [1.0, 0.48333333333333334]
21 : [1.0, 0.45]
22 : [1.0, 0.5]
23 : [1.0, 0.4666666666666667]
24 : [1.0, 0.5166666666666667]
25 : [1.0, 0.4666666666666667]
26 : [1.0, 0.48333333333333334]
27 : [1.0, 0.45]
```

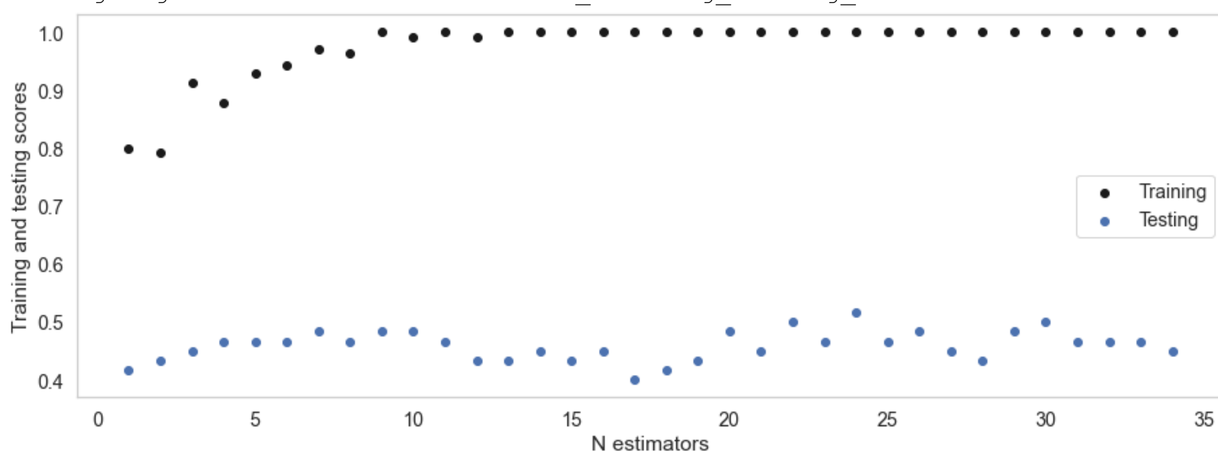
```
28 : [1.0, 0.43333333333333335]
29 : [1.0, 0.48333333333333334]
30 : [1.0, 0.5]
31 : [1.0, 0.4666666666666667]
32 : [1.0, 0.4666666666666667]
33 : [1.0, 0.4666666666666667]
34 : [1.0, 0.45]
```

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

In [149…

```
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 [150…

```
### 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:  0.5899280575539568
```

In [151…

```
### 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:  0.539568345323741
```

`MultinomialNB` performs better than the others.

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

In [152…

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


y_pred_nb = multinomNB_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: 0.45
[[21  2  2  3]
 [ 0  2  3  1]
 [ 8  4  4  0]
 [ 8  0  2  0]]
                  precision    recall  f1-score   support

         Average       0.57      0.75      0.65        28
       Excellent       0.25      0.33      0.29         6
            Good       0.36      0.25      0.30        16
Not Satisfactory       0.00      0.00      0.00        10

        accuracy                           0.45        60
       macro avg       0.30      0.33      0.31        60
    weighted avg       0.39      0.45      0.41        60
```

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

In [153…

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold


cv_mult_nb = KFold(n_splits=5, shuffle=True, random_state=42)
cross_val_score(multinomNB_clf, X_train, y_train, cv=cv_mult_nb, scoring="accu
```

Out[153…

```
array([0.64285714, 0.53571429, 0.25      , 0.39285714, 0.48148148])
```

In [154…
```
scores = cross_val_score(multinomNB_clf, X_test, y_test, cv=cv_mult_nb, scorin
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.533 (0.180)
```

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

In [155…
```
scores = cross_val_score(multinomNB_clf, X_test, y_test, cv=4, scoring="accura
print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
```

```
Accuracy: 0.483 (0.073)
```

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 [156…
```
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)  142.328351
1            Total Internet Usage(hrs/day)   62.396491
3     Duration Of Internet Usage(In Years)   26.387671
13                 Purpose Of Internet Use   20.515724
14                        Browsing Purpose   19.334110
15    Priority Of Learning On The Internet   13.131686
6          Effectiveness Of Internet Usage   10.802870
17  Internet Usage For Educational Purpose    8.195196
16                                 Webinar    5.050512
4                                   Gender    3.286463
```

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 [157…

```
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)
```

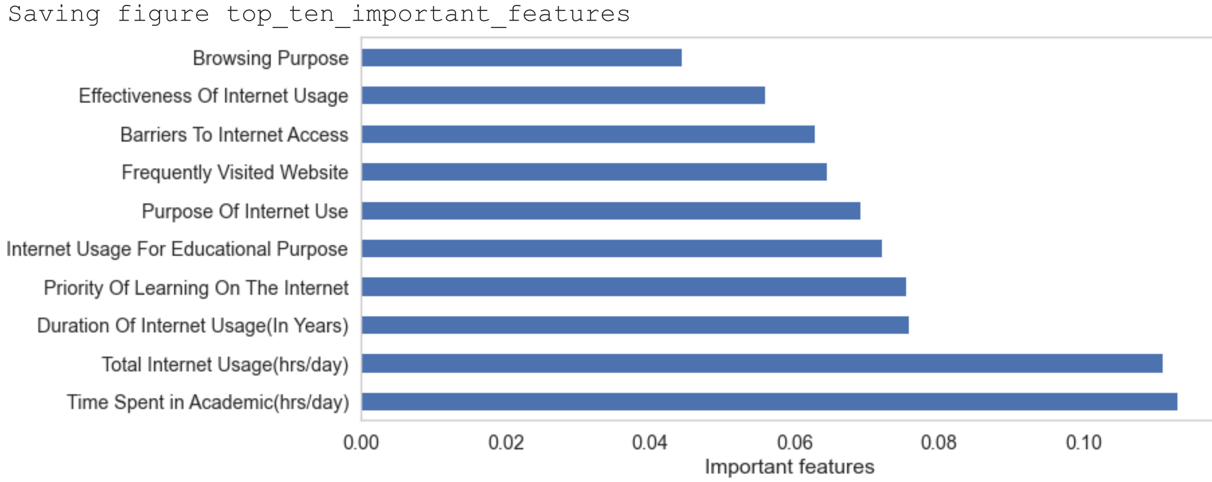
```
[0.02515274 0.11098995 0.11302889 0.07574717 0.033249   0.06440715
 0.05590525 0.03717398 0.01978781 0.02187141 0.02435687 0.03584619
 0.03521724 0.06914282 0.04434116 0.07546094 0.0235558  0.07205592
 0.06270969]
```

Let's plot the top 10 most important features.

In [158…

```
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
```

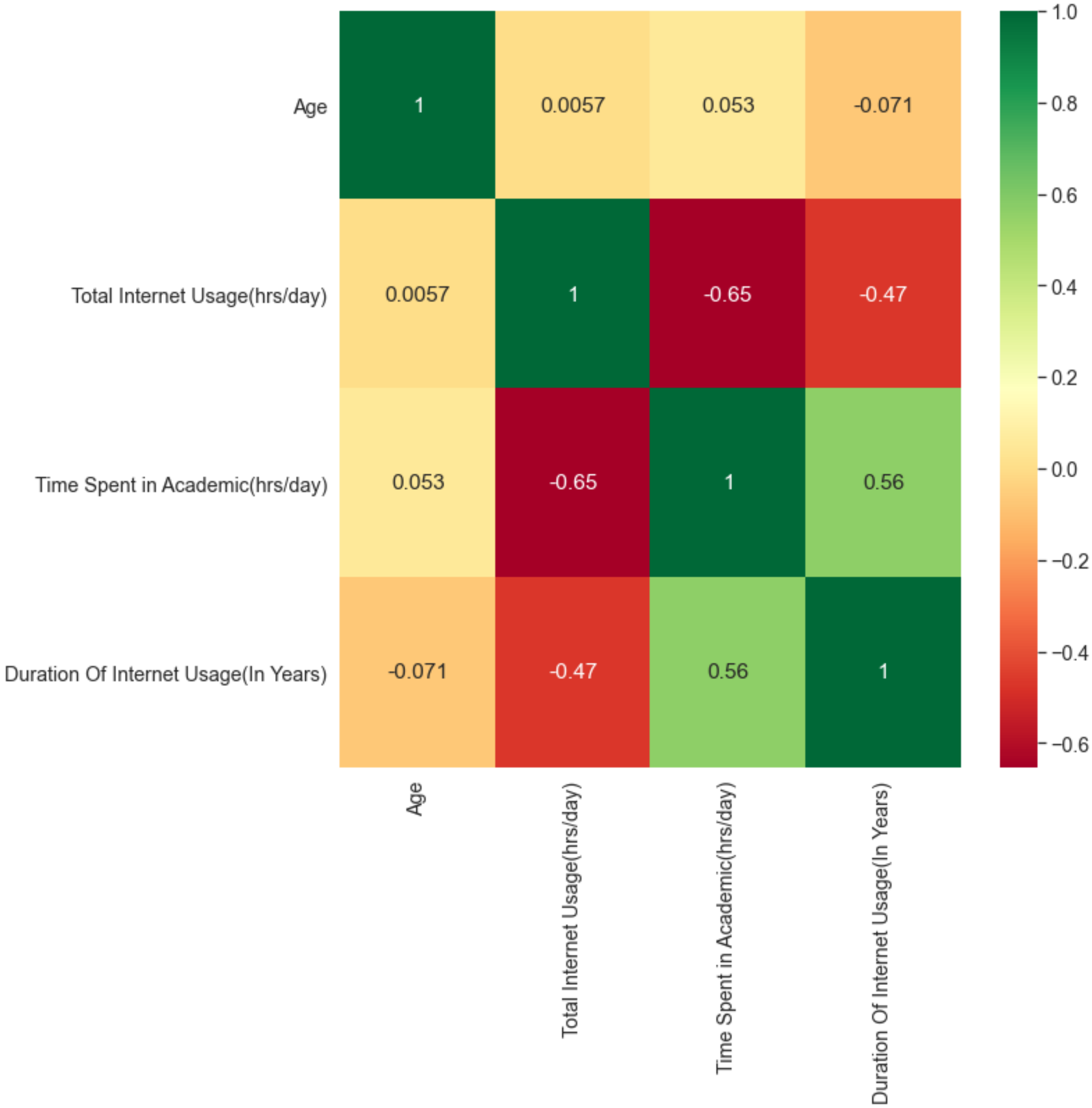


Correlation Matrix with Heatmap

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 [159…

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

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

#plot heat map
g=sns.heatmap(college_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.
2. **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.
3. **BayesSearchCV** - Bayesian Optimization of model hyperparameters provided by the Scikit-Optimize library.
4. **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 `MultinomialNB`. Let's try and optimize the `MultinomialNB` algorithm more to get a better result. First let's see the parameters that we'll try and tune in the `MultinomialNB`.

In [192…
```
from sklearn.naive_bayes import MultinomialNB


multinomNB_clf = MultinomialNB()

multinomNB_clf.fit(X_train, y_train)

multinomNB_clf.get_params().keys()
```

Out[192…
```
dict_keys(['alpha', 'class_prior', 'fit_prior'])
```

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

In [193…
```
grid_params = {
  'alpha': np.linspace(0.5, 100.5, 100),
  'fit_prior': [True, False],
}

print(grid_params)
```

```
{'alpha': array([  0.5       ,   1.51010101,   2.52020202,   3.53030303,
         4.54040404,   5.55050505,   6.56060606,   7.57070707,
         8.58080808,   9.59090909,  10.6010101 ,  11.61111111,
        12.62121212,  13.63131313,  14.64141414,  15.65151515,
        16.66161616,  17.67171717,  18.68181818,  19.69191919,
        20.7020202 ,  21.71212121,  22.72222222,  23.73232323,
        24.74242424,  25.75252525,  26.76262626,  27.77272727,
```

```
28.78282828,  29.79292929,  30.8030303 ,  31.81313131,
32.82323232,  33.83333333,  34.84343434,  35.85353535,
36.86363636,  37.87373737,  38.88383838,  39.89393939,
40.9040404 ,  41.91414141,  42.92424242,  43.93434343,
44.94444444,  45.95454545,  46.96464646,  47.97474747,
48.98484848,  49.99494949,  51.00505051,  52.01515152,
53.02525253,  54.03535354,  55.04545455,  56.05555556,
57.06565657,  58.07575758,  59.08585859,  60.0959596 ,
61.10606061,  62.11616162,  63.12626263,  64.13636364,
65.14646465,  66.15656566,  67.16666667,  68.17676768,
69.18686869,  70.1969697 ,  71.20707071,  72.21717172,
73.22727273,  74.23737374,  75.24747475,  76.25757576,
77.26767677,  78.27777778,  79.28787879,  80.2979798 ,
81.30808081,  82.31818182,  83.32828283,  84.33838384,
85.34848485,  86.35858586,  87.36868687,  88.37878788,
89.38888889,  90.3989899 ,  91.40909091,  92.41919192,
93.42929293,  94.43939394,  95.44949495,  96.45959596,
97.46969697,  98.47979798,  99.48989899, 100.5       ]), 'fit_prior':
```

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

In [194…

```
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=5, n_repeats=3, random_state=1)

# define the search
gs_multinom_nb = GridSearchCV(multinomNB_clf, param_grid=grid_params, scoring=

gs_multinom_nb.fit(X_train, y_train)

gs_multinom_nb.best_params_
```

Out[194…

```
{'alpha': 44.94444444444445, 'fit_prior': False}
```

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

In [195…

```
gs_multinom_nb.score(X_train, y_train)
```

Out[195…

```
0.5539568345323741
```

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

In [196…

```
y_pred_gs_multinom = gs_multinom_nb.predict(X_test)

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


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

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.4666666666666667
[[26  1  1  0]
 [ 4  2  0  0]
```

```
 [13  3  0  0]
 [10  0  0  0]]
                  precision    recall  f1-score   support

         Average       0.49      0.93      0.64        28
       Excellent       0.33      0.33      0.33         6
            Good       0.00      0.00      0.00        16
Not Satisfactory       0.00      0.00      0.00        10

        accuracy                           0.47        60
       macro avg       0.21      0.32      0.24        60
    weighted avg       0.26      0.47      0.33        60

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))
```

Hyperparameter Optimization using `RandomizedSearchCV`

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

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 [197…
```python
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform

# define evaluation
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1)

rs_multinom_nb = RandomizedSearchCV(multinomNB_clf, grid_params, scoring='accu

rs_multinom_nb.fit(X_train, y_train)

rs_multinom_nb.best_params_
```

Out[197…
```
{'fit_prior': True, 'alpha': 31.813131313131315}
```

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

In [198…
```python
rs_multinom_nb.score(X_train, y_train)
```

Out[198…
```
0.5467625899280576
```

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

In [199…

```
y_pred_rs_multinom = rs_multinom_nb.predict(X_test)

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


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

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.4666666666666667
[[26  2  0  0]
 [ 4  2  0  0]
 [13  3  0  0]
 [10  0  0  0]]
                  precision    recall  f1-score   support

         Average       0.49      0.93      0.64        28
       Excellent       0.29      0.33      0.31         6
            Good       0.00      0.00      0.00        16
Not Satisfactory       0.00      0.00      0.00        10

        accuracy                           0.47        60
       macro avg       0.19      0.32      0.24        60
    weighted avg       0.26      0.47      0.33        60

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))
```

Hyperparameter Optimization using `BayesSearchCV`

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

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 [201…

```
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_multinom_nb = BayesSearchCV(estimator=multinomNB_clf, search_spaces=grid_pa

# perform the search
bs_multinom_nb.fit(X, y)

# report the best result
print(bs_multinom_nb.best_score_)
print(bs_multinom_nb.best_params_)
```

```
E:\Users\MSI\anaconda3\lib\site-packages\skopt\optimizer\optimizer.py:449: Use
rWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
0.5278070175438596
OrderedDict([('alpha', 48.984848484848484), ('fit_prior', True)])
```

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

In [202…

```
bs_multinom_nb.score(X_train, y_train)
```

Out[202…

```
0.5611510791366906
```

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

In [203…

```
y_pred_bs_multinom = bs_multinom_nb.predict(X_test)

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


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

print(conf_mat)
print(class_report)
```

```
Accuracy: 0.5166666666666667
[[26  1  1  0]
 [ 4  2  0  0]
 [13  0  3  0]
 [10  0  0  0]]
                  precision    recall  f1-score   support

         Average       0.49      0.93      0.64        28
       Excellent       0.67      0.33      0.44         6
            Good       0.75      0.19      0.30        16
Not Satisfactory       0.00      0.00      0.00        10

        accuracy                           0.52        60
       macro avg       0.48      0.36      0.35        60
    weighted avg       0.50      0.52      0.42        60

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.
```

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 [204…

```python
# 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[204…

```
array([0, 3, 3, 2, 3, 2, 0, 0, 0, 2, 1, 1, 0, 0, 2, 0, 0, 0, 3, 2, 0, 2,
       0, 1, 1, 0, 1, 1, 0, 3, 3, 0, 3, 1, 0, 0, 0, 0, 0, 3, 2, 0, 0, 2,
       3, 2, 2, 0, 0, 0, 3, 0, 2, 1, 3, 0, 1, 1, 3, 0, 2, 3, 3, 1, 1, 0,
       0, 2, 3, 0, 2, 0, 3, 2, 3, 3, 0, 2, 0, 0, 0, 2, 0, 0, 2, 3, 2, 0,
       1, 0, 1, 0, 0, 3, 0, 1, 0, 3, 1, 0, 0, 3, 0, 3, 0, 1, 0, 0, 0, 2,
       2, 2, 3, 0, 0, 0, 0, 1, 1, 0, 2, 3, 3, 2, 3, 2, 2, 0, 0, 2, 2, 1,
       0, 0, 0, 0, 3, 0, 0])
```

In [205…

```python
y_train.head(20)
```

Out[205…

```
113             Average
71     Not Satisfactory
180    Not Satisfactory
76                 Good
12     Not Satisfactory
21                 Good
```

```
26               Average
157              Average
108              Average
160                 Good
190            Excellent
88             Excellent
178              Average
40               Average
139                 Good
77               Average
58               Average
25               Average
16      Not Satisfactory
78                  Good
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 [206…

```python
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_college.py')
```

```
Generation 1 - Current best internal CV score: 0.5835164835164834

Generation 2 - Current best internal CV score: 0.5835164835164834

Generation 3 - Current best internal CV score: 0.5835164835164834

Generation 4 - Current best internal CV score: 0.5835164835164834

Generation 5 - Current best internal CV score: 0.5835164835164834

Generation 6 - Current best internal CV score: 0.5835164835164834

Generation 7 - Current best internal CV score: 0.5835164835164834

Generation 8 - Current best internal CV score: 0.5835164835164834

Generation 9 - Current best internal CV score: 0.5972527472527472

Generation 10 - Current best internal CV score: 0.5972527472527472

Generation 11 - Current best internal CV score: 0.5972527472527472
```

```
Generation 12 - Current best internal CV score: 0.5972527472527472

Generation 13 - Current best internal CV score: 0.5978021978021978

Generation 14 - Current best internal CV score: 0.5978021978021978

Generation 15 - Current best internal CV score: 0.5978021978021978

Best pipeline: GradientBoostingClassifier(SelectPercentile(input_matrix, perce
ntile=2), learning_rate=0.1, max_depth=8, max_features=0.25, min_samples_leaf=
5, min_samples_split=6, n_estimators=100, subsample=0.8500000000000001)
```

Genetic algorithm showed us that the most optimized algorithm is the `GradientBoostingClassifier` with the following parameter :

`GradientBoostingClassifier(SelectPercentile(input_matrix, percentile=2), learning_rate=0.1, max_depth=8, max_features=0.25, min_samples_leaf=5, min_samples_split=6, n_estimators=100, subsample=0.8500000000000001)` `0.48333333333333334`

Let's fit this algorithm to our dataset and check the training score.

In [207…

```python
import numpy as np
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline, make_union
from tpot.builtins import StackingEstimator
from tpot.export_utils import set_param_recursive


# Average CV score on the training set was: 0.7714285714285715
exported_pipeline = make_pipeline(
    StackingEstimator(estimator=ExtraTreesClassifier(bootstrap=False, criteri
    ExtraTreesClassifier(bootstrap=False, criterion="entropy", max_features=0
)
# 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)

score = exported_pipeline.score(X_train, y_train_n)
print("Training score: ", score)
```

```
Training score:  0.7985611510791367
```

Let's check the accuracy on the test set and check the confusion matrix, precision, recall and f1 scores.

In [208…

```
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: 0.5
[[25  2  0  1]
 [ 2  4  0  0]
 [ 9  6  1  0]
 [ 9  1  0  0]]
              precision    recall  f1-score   support

           0       0.56      0.89      0.68        28
           1       0.31      0.67      0.42         6
           2       1.00      0.06      0.12        16
           3       0.00      0.00      0.00        10

    accuracy                           0.50        60
   macro avg       0.47      0.41      0.31        60
weighted avg       0.56      0.50      0.39        60
```

Finally, let's perform `KFold` cross validation.

In [209…

```
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(sc

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: 0.498 (0.111)
Testing Accuracy On KFold Cross Validation: 0.467 (0.208)
```

This model givess us a 46.7% accuracy on KFold cross validation.

In [ ]: