In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from IPython import get_ipython
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
data = pd.read_csv('online_education.csv')
```

In [3]:

```
data.head()
```

Out[3]:

| | Education Level | Institution Type | Gender | Age | Device | IT Student | Location | Financial Condition | Internet Type | Network Type | F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | University | Private | Male | 23 | Tab | No | Town | Mid | Wifi | 4G | M |
| **1** | University | Private | Female | 23 | Mobile | No | Town | Mid | Mobile Data | 4G | M |
| **2** | College | Public | Female | 18 | Mobile | No | Town | Mid | Wifi | 4G | M |
| **3** | School | Private | Female | 11 | Mobile | No | Town | Mid | Mobile Data | 4G | M |
| **4** | School | Private | Female | 18 | Mobile | No | Town | Poor | Mobile Data | 3G | |

In [4]:

```
data.tail()
```

Out[4]:

| | Education Level | Institution Type | Gender | Age | Device | IT Student | Location | Financial Condition | Internet Type | Networl Type |
|---|---|---|---|---|---|---|---|---|---|---|
| **1200** | College | Private | Female | 18 | Mobile | No | Town | Mid | Wifi | 4C |
| **1201** | College | Private | Female | 18 | Mobile | No | Rural | Mid | Wifi | 4C |
| **1202** | School | Private | Male | 11 | Mobile | No | Town | Mid | Mobile Data | 3C |
| **1203** | College | Private | Female | 18 | Mobile | No | Rural | Mid | Wifi | 4C |
| **1204** | School | Private | Female | 11 | Mobile | No | Town | Poor | Mobile Data | 3C |

In [5]:

```
1 data.shape
```

Out[5]:

```
(1205, 11)
```

In [6]:

```
1 data.columns
```

Out[6]:

```
Index(['Education Level', 'Institution Type', 'Gender', 'Age', 'Device',
       'IT Student', 'Location', 'Financial Condition', 'Internet Type',
       'Network Type', 'Flexibility Level'],
      dtype='object')
```

In [7]:

```
1 data.duplicated().sum()
```

Out[7]:

```
980
```

In [8]:

```
1 data.isnull().sum()
```

Out[8]:

```
Education Level        0
Institution Type       0
Gender                 0
Age                    0
Device                 0
IT Student             0
Location               0
Financial Condition    0
Internet Type          0
Network Type           0
Flexibility Level      0
dtype: int64
```

In [9]:

```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1205 entries, 0 to 1204
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Education Level      1205 non-null   object
 1   Institution Type     1205 non-null   object
 2   Gender               1205 non-null   object
 3   Age                  1205 non-null   int64 
 4   Device               1205 non-null   object
 5   IT Student           1205 non-null   object
 6   Location             1205 non-null   object
 7   Financial Condition  1205 non-null   object
 8   Internet Type        1205 non-null   object
 9   Network Type         1205 non-null   object
 10  Flexibility Level    1205 non-null   object
dtypes: int64(1), object(10)
memory usage: 103.7+ KB
```

In [10]:

```
1 data.describe()
```

Out[10]:

| | Age |
|---|---|
| **count** | 1205.000000 |
| **mean** | 17.065560 |
| **std** | 5.830369 |
| **min** | 9.000000 |
| **25%** | 11.000000 |
| **50%** | 18.000000 |
| **75%** | 23.000000 |
| **max** | 27.000000 |

In [11]:

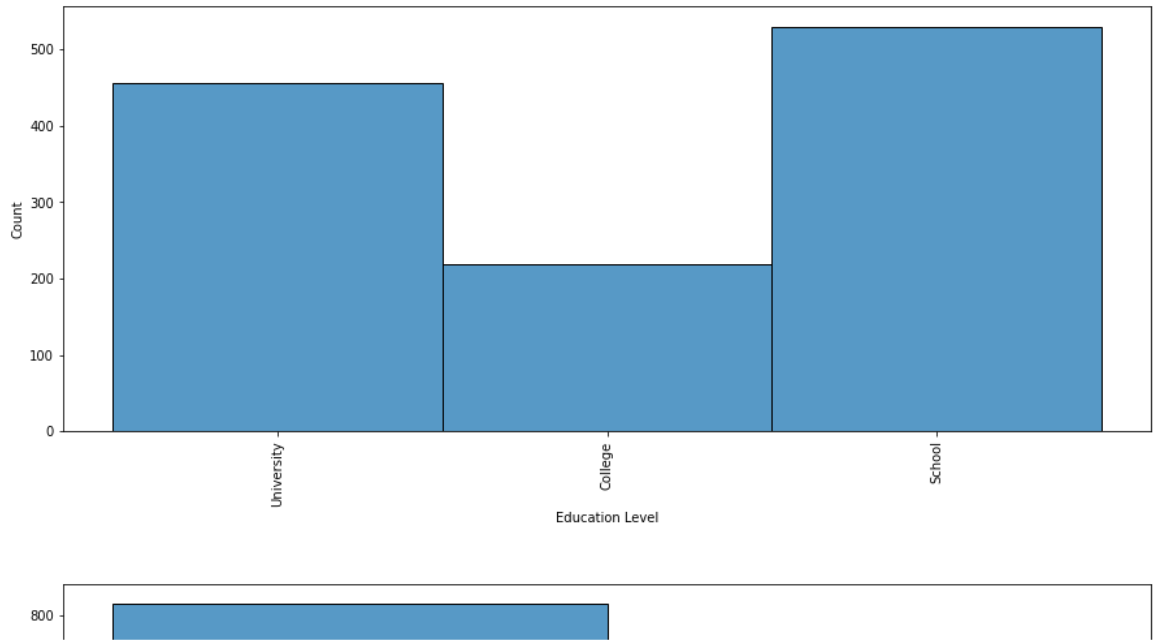
```
1 data.nunique()
```

Out[11]:

```
Education Level         3
Institution Type        2
Gender                  2
Age                     6
Device                  3
IT Student              2
Location                2
Financial Condition     3
Internet Type           2
Network Type            3
Flexibility Level       3
dtype: int64
```

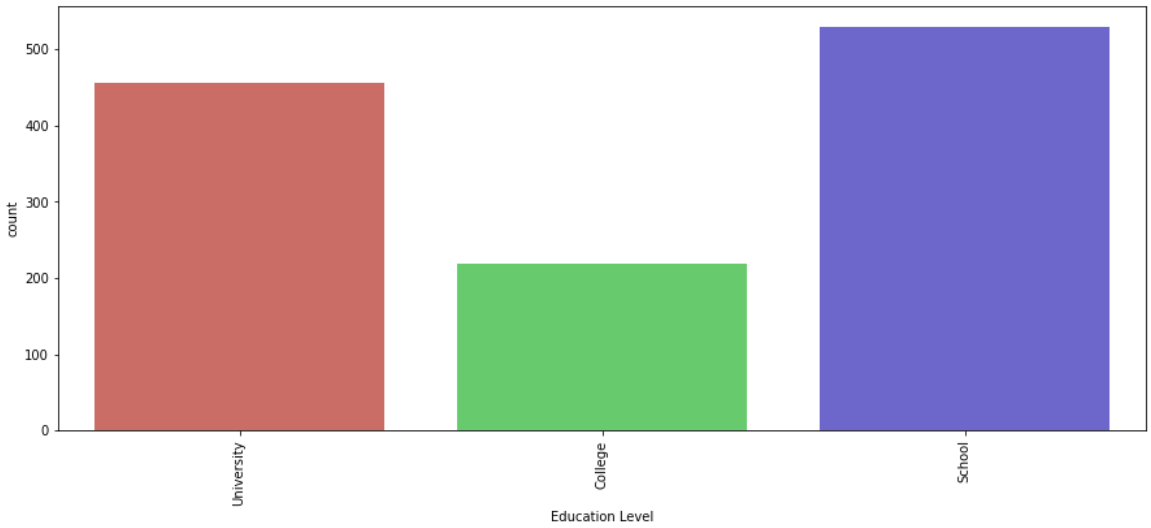
In [12]:

```
1 for i in data.columns:
2     plt.figure(figsize=(15,6))
3     sns.histplot(data[i], bins=10)
4     plt.xticks(rotation = 90)
5     plt.show()
```



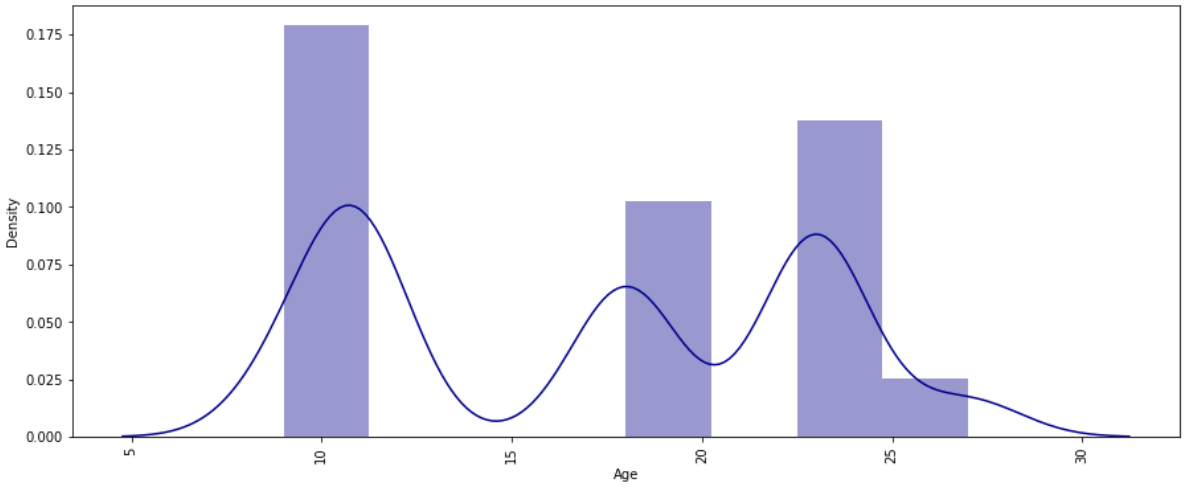
In [13]:

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



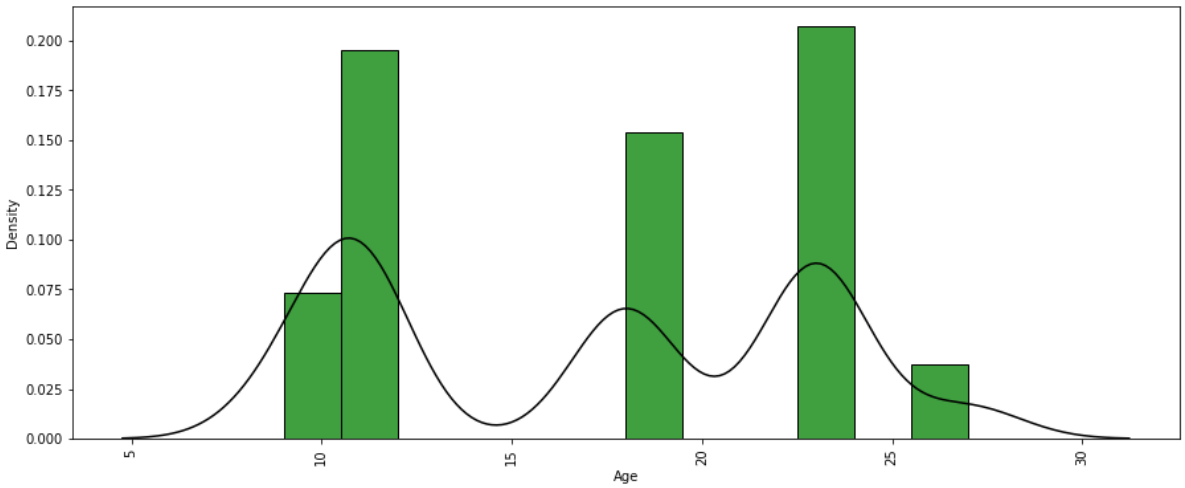

In [14]:

```
plt.figure(figsize=(15,6))
sns.distplot(data['Age'],kde = True,color = 'Darkblue')
plt.xticks(rotation = 90)
plt.show()
```



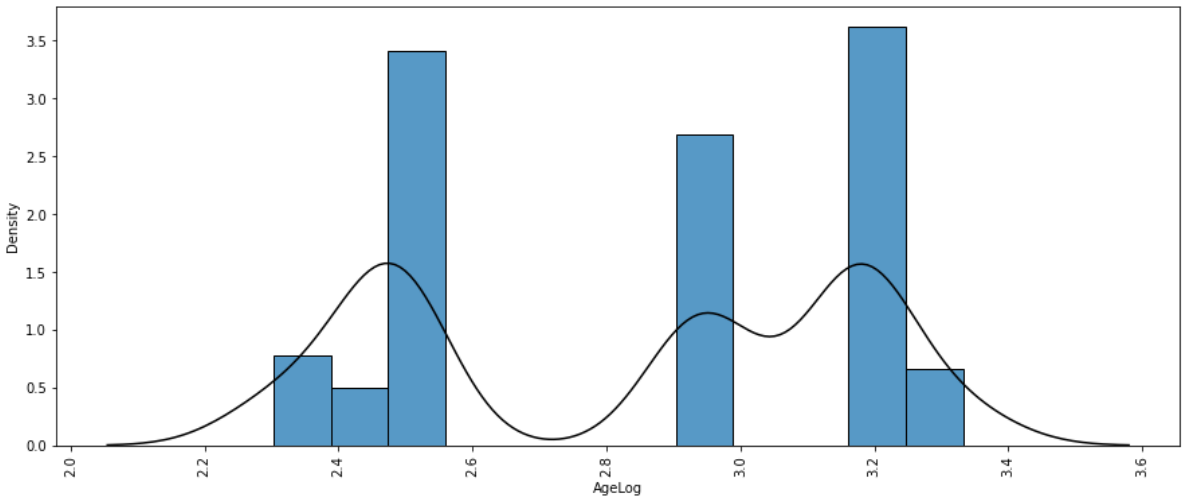
In [15]:

```
plt.figure(figsize=(15,6))
sns.histplot(data["Age"], stat='density',color='green')
sns.kdeplot(data["Age"], color='black')
plt.xticks(rotation = 90)
plt.show()
```



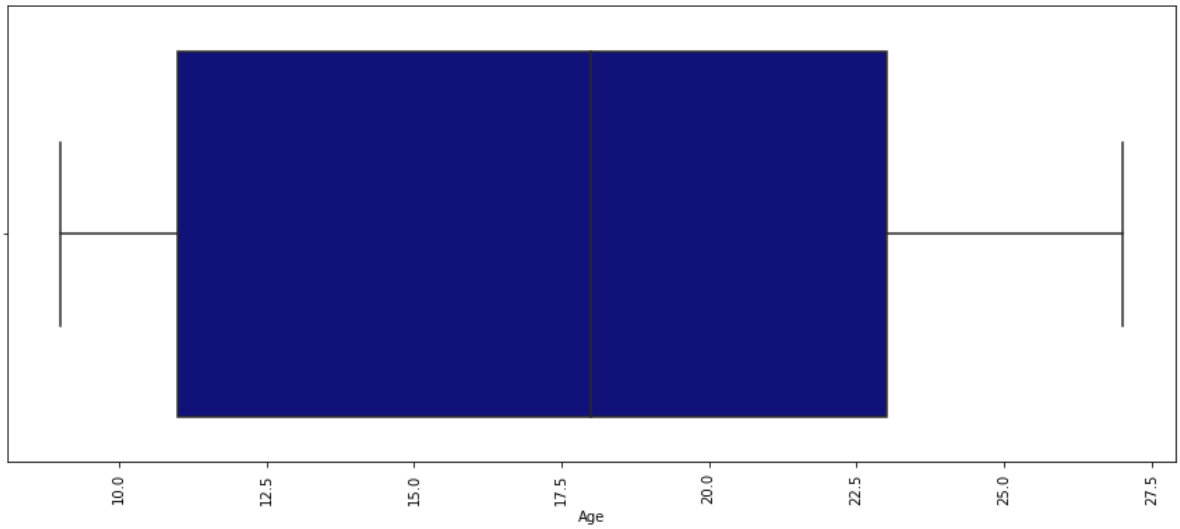

In [16]:

```
plt.figure(figsize=(15,6))
data['AgeLog'] = np.log(data['Age']+1)
sns.histplot(data["AgeLog"], stat='density')
sns.kdeplot(data["AgeLog"], color='black')
plt.xticks(rotation = 90)
plt.show()
```



In [17]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data['Age'],color = 'Darkblue')
plt.xticks(rotation = 90)
plt.show()
```




In [18]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [19]:

```
FlexibilityLevel = le.fit_transform(data['Flexibility Level'])
data['Flexibility Level'] = FlexibilityLevel
```

In [20]:

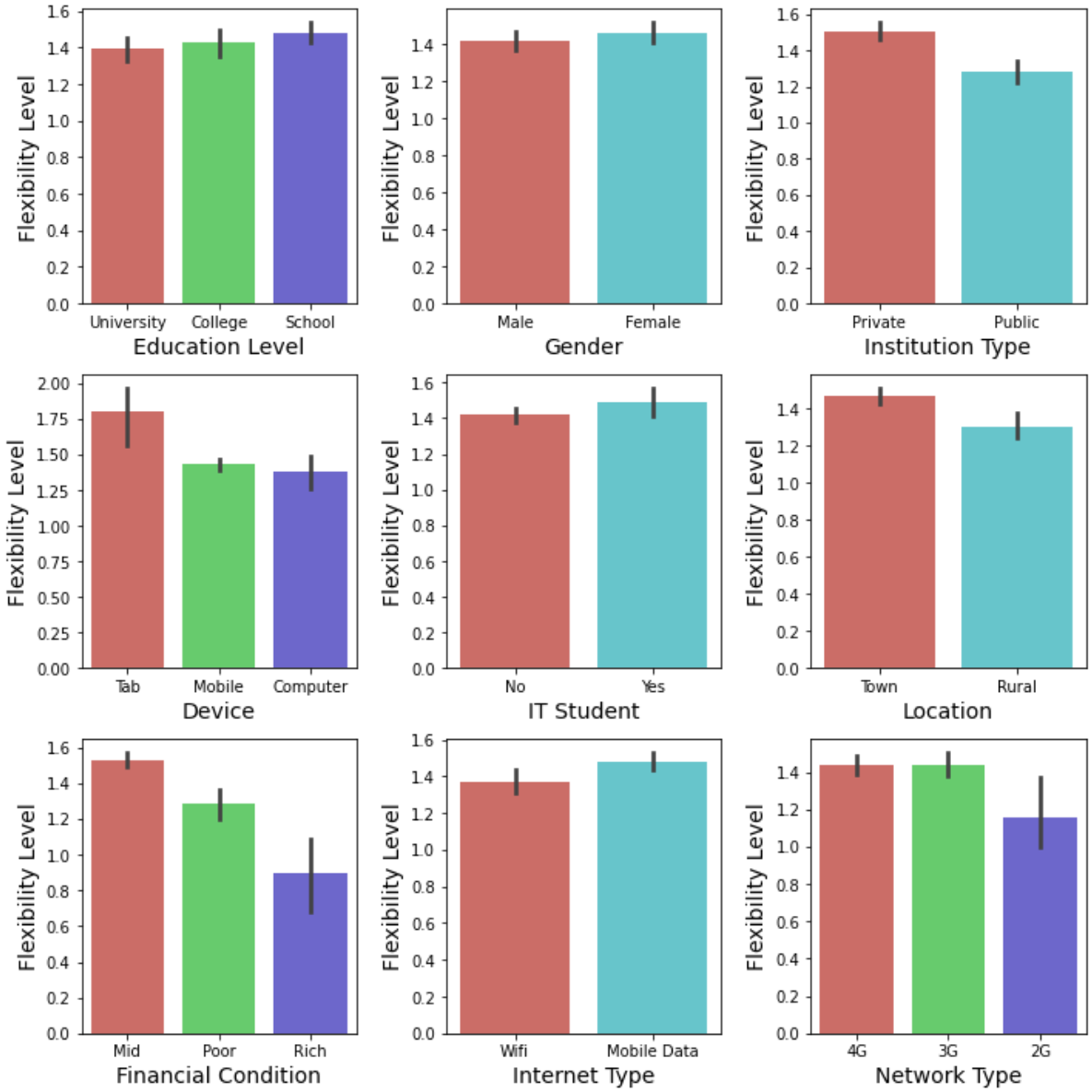
```
categorical_features = ['Education Level','Gender','Institution Type',
                        'Device','IT Student','Location',
                        'Financial Condition','Internet Type',
                        'Network Type']
```

In [21]:

```
fig, axes = plt.subplots(3, 3, figsize=(10,10))
axes = [ax for axes_row in axes for ax in axes_row]
target = 'Flexibility Level'

for i, c in enumerate(categorical_features):
    sns.barplot(data[c], data[target], ax=axes[i],
                palette = 'hls')
    axes[i].set_ylabel('Flexibility Level', fontsize=14)
    axes[i].set_xlabel(c, fontsize=14)

plt.tight_layout()
plt.show()
```



In [22]:

```
EducationLevel= le.fit_transform(data['Education Level'])
Gender = le.fit_transform(data['Gender'])
InstitutionType = le.fit_transform(data['Institution Type'])
Device = le.fit_transform(data['Device'])
ITStudent= le.fit_transform(data['IT Student'])
Location = le.fit_transform(data['Location'])
FinancialCondition = le.fit_transform(data['Financial Condition'])
InternetType = le.fit_transform(data['Internet Type'])
NetworkType = le.fit_transform(data['Network Type'])
```

In [23]:

```
data['Education Level'] = EducationLevel
data['Gender'] = Gender
data['Institution Type'] = InstitutionType
data['Device'] = Device
data['IT Student'] = ITStudent
data['Location'] = Location
data['Financial Condition'] = FinancialCondition
data['Internet Type'] = InternetType
data['Network Type'] = NetworkType
```

In [24]:

```
data.head()
```

Out[24]:

| | Education Level | Institution Type | Gender | Age | Device | IT Student | Location | Financial Condition | Internet Type | Network Type | F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2 | 0 | 1 | 23 | 2 | 0 | 1 | 0 | 1 | 2 | |
| **1** | 2 | 0 | 0 | 23 | 1 | 0 | 1 | 0 | 0 | 2 | |
| **2** | 0 | 1 | 0 | 18 | 1 | 0 | 1 | 0 | 1 | 2 | |
| **3** | 1 | 0 | 0 | 11 | 1 | 0 | 1 | 0 | 0 | 2 | |
| **4** | 1 | 0 | 0 | 18 | 1 | 0 | 1 | 1 | 0 | 1 | |

In [25]:

```
x = data.drop('Flexibility Level',axis=1)
y = data['Flexibility Level']
```

In [26]:

```
x.shape
```

Out[26]:

```
(1205, 11)
```

In [27]:

```
y.shape
```

Out[27]:

```
(1205,)
```

In [28]:

```
from sklearn.preprocessing import StandardScaler
```

In [29]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.30,
                                                    random_state=42)
```

In [30]:

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

In [31]:

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

Out[31]:

```
DecisionTreeClassifier()
```

In [32]:

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

In [33]:

```
print("Training Accuracy :", model.score(X_train, y_train))
print("Testing Accuracy :", model.score(X_test, y_test))
```

```
Training Accuracy : 0.8623962040332147
Testing Accuracy : 0.8011049723756906
```

In [34]:

```
1 from sklearn.metrics import classification_report
```

In [35]:

```
1 print(classification_report(y_test, y_pred))
```

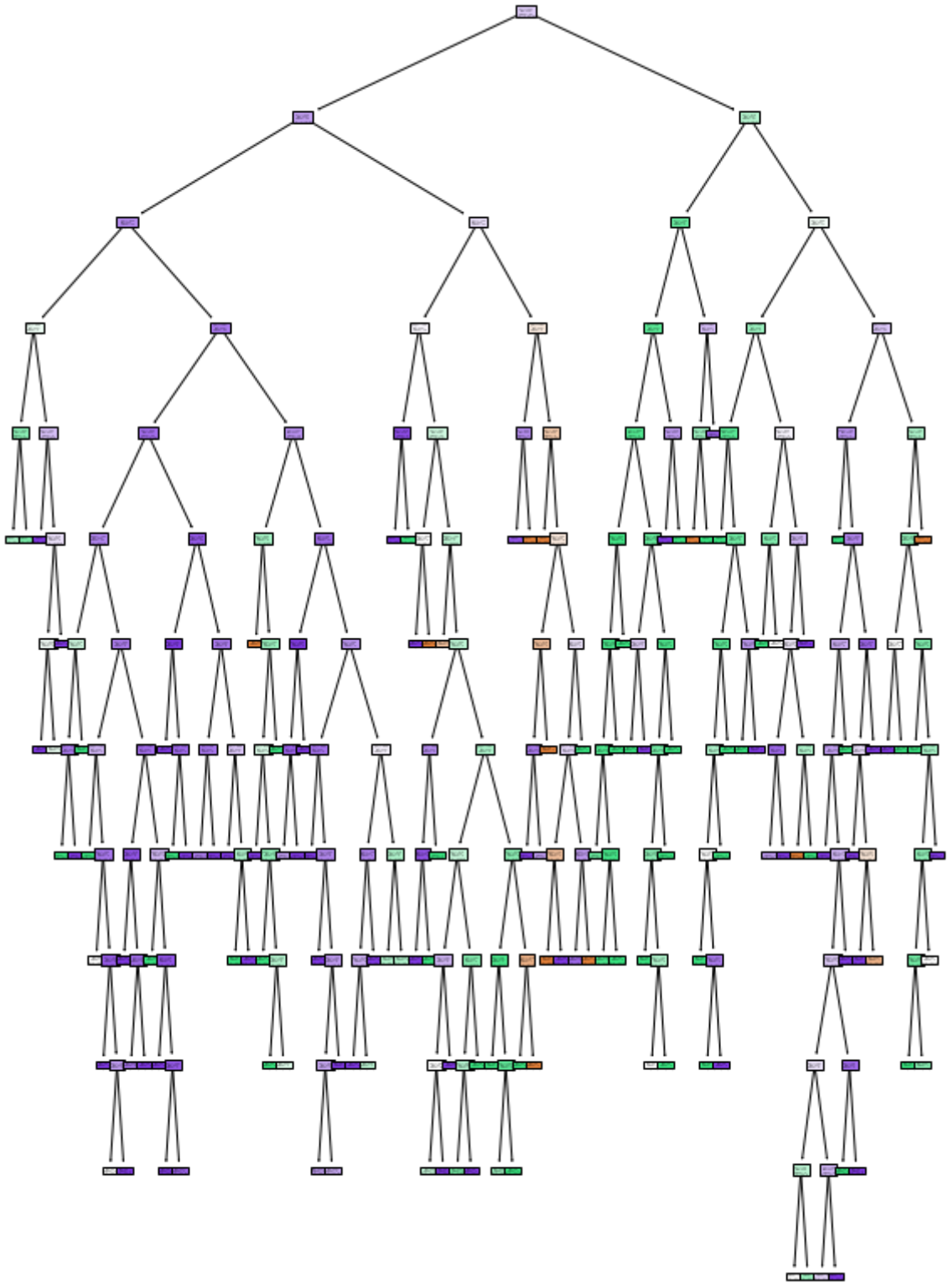
```
              precision    recall  f1-score   support

           0       0.85      0.56      0.68        39
           1       0.82      0.82      0.82       154
           2       0.78      0.84      0.81       169

    accuracy                           0.80       362
   macro avg       0.82      0.74      0.77       362
weighted avg       0.80      0.80      0.80       362
```

In [36]:

```
from sklearn import tree
plt.figure(figsize=(10,15))
tree.plot_tree(model,filled=True)
plt.show()
```



In [37]:

```
1 model.feature_importances_
```

Out[37]:

```
array([0.12189951, 0.09626619, 0.1171519 , 0.02145348, 0.04924518,
       0.06726136, 0.05672186, 0.12951326, 0.06101387, 0.12435672,
       0.15511667])
```

In [38]:

```
1 from sklearn.ensemble import RandomForestClassifier
2 classifier= RandomForestClassifier(n_estimators= 10,
3                                    criterion="entropy")
```

In [39]:

```
classifier.fit(X_train, y_train)
```

Out[39]:

```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [40]:

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

In [41]:

```
print("Training Accuracy :", classifier.score(X_train, y_train))
print("Testing Accuracy :", classifier.score(X_test, y_test))
```

```
Training Accuracy : 0.8588374851720048
Testing Accuracy : 0.787292817679558
```

In [42]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.54      0.69        39
           1       0.78      0.84      0.81       154
           2       0.77      0.80      0.78       169

    accuracy                           0.79       362
   macro avg       0.84      0.72      0.76       362
weighted avg       0.80      0.79      0.78       362
```

In [43]:

```
classifier.feature_importances_
```

Out[43]:

```
array([0.07576819, 0.08319807, 0.11572646, 0.0980481 , 0.05521592,
       0.06144166, 0.06213362, 0.18141846, 0.07569019, 0.08614288,
       0.10521646])
```

In [44]:

```
grid_param = {
    'criterion': ['gini', 'entropy'],
    'max_depth' : range(2,32,1),
    'min_samples_leaf' : range(1,10,1),
    'min_samples_split': range(2,10,1),
    'splitter' : ['best', 'random']
}
```

In [46]:

```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=model,
                     param_grid=grid_param,
                     cv=5,
                    n_jobs =2, verbose=1)
```

In [47]:

```
grid_search.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 8640 candidates, totalling 43200 fits
```

Out[47]:

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=2,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']},
             verbose=1)
```

In [48]:

```
best_parameters = grid_search.best_params_
print(best_parameters)
```

```
{'criterion': 'gini', 'max_depth': 16, 'min_samples_leaf': 1, 'min_samples
_split': 2, 'splitter': 'random'}
```

In [49]:

```
grid_search.best_score_
```

Out[49]:

```
0.7936038320653704
```

In [51]:

```
clf = DecisionTreeClassifier(criterion = 'gini', max_depth = 25,
                             min_samples_leaf= 1, min_samples_split= 2,
                             splitter ='best')
clf.fit(X_train,y_train)
```

Out[51]:

```
DecisionTreeClassifier(max_depth=25)
```

In [52]:

```
ac2_clf= clf.score(X_test ,y_test)
ac2_clf
```

Out[52]:

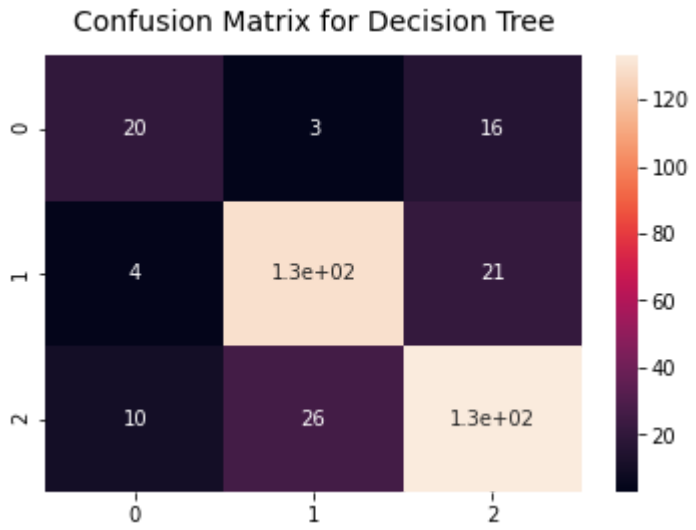
```
0.7790055248618785
```

In [53]:

```
from sklearn.metrics import confusion_matrix
y_pred_clf = clf.predict(X_test)
cf_matrix = confusion_matrix(y_test, y_pred_clf)
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for Decision Tree", fontsize=14,
          fontname="DejaVu Sans", y=1.03)
```

Out[53]:

```
Text(0.5, 1.03, 'Confusion Matrix for Decision Tree')
```




In [55]:

```
y_pred_rf = classifier.predict(X_test)
```

In [58]:

```
from sklearn.metrics import accuracy_score
ac_rf=accuracy_score(y_test,y_pred)
ac_rf
```

Out[58]:

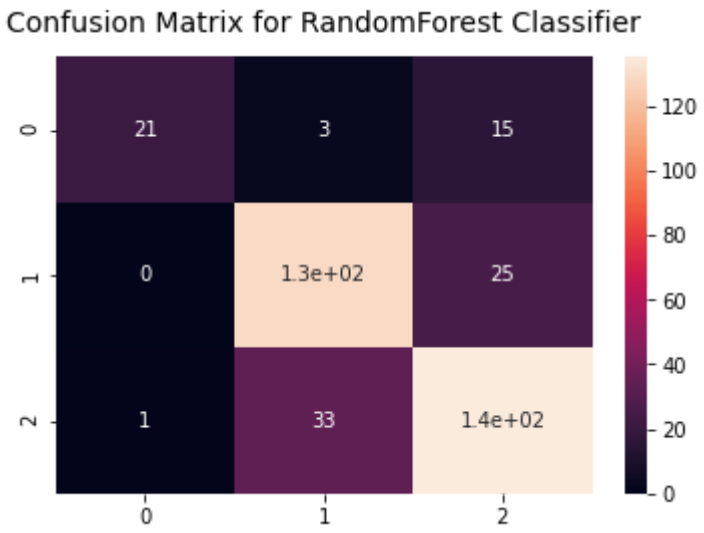
```
0.787292817679558
```

In [56]:

```
cf_matrix = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cf_matrix, annot=True)
plt.title("Confusion Matrix for RandomForest Classifier",
          fontsize=14, fontname="DejaVu Sans", y=1.03)
```

Out[56]:

```
Text(0.5, 1.03, 'Confusion Matrix for RandomForest Classifier')
```




In [59]:

```
plt.bar(['Decision Tree','Random Forest'],[ac2_clf,ac_rf])
plt.xlabel("Algorithms")
plt.ylabel("Accuracy")
plt.show()
```