In [1]:
```
"""
Author: Chidura Santosh

Date: 12-April-2019

"""
```

Out[1]:
```
'\nAuthor: Chidura Santosh\n\nDate: 12-April-2019\n\n'
```

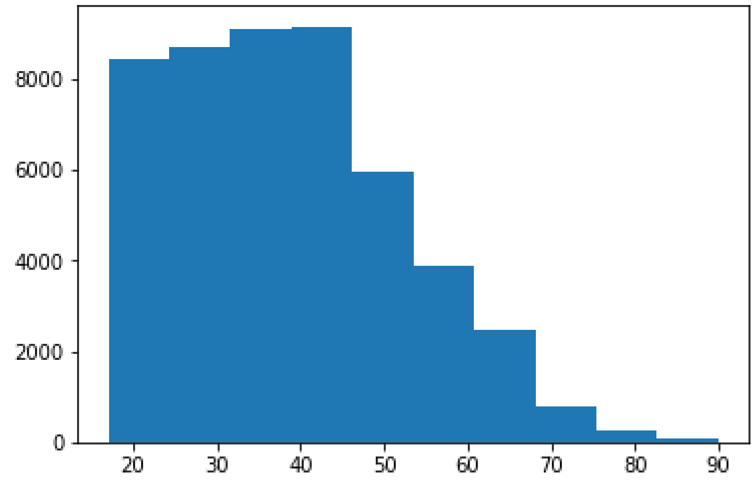
In [2]:
```
#Importing Required Libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [3]:
```
# Reading Train and Tets data from given source
train_set =pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data', header = None)
test_set= pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.test',skiprows=1 ,header = Non
col_labels = ['age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship',
train_set.columns = col_labels
test_set.columns = col_labels
```

In [4]:
```
# Adding Training and test data into one data frame df
df=pd.concat([train_set,test_set],axis=0)
df.head()
original_df=df # Keeping copy of original data frame
```

In [5]:

```
# Plotting Histigram of Age feature
plt.hist(df['age']);
```



In [6]:

```
# Making target column as categorical value as for >50K salaris as 1 and <=50K as 0
df['Income'] = df['income'].apply(lambda x: 1 if x==' >50K' else 0)
df.drop('income',axis=1,inplace=True)
```

In [7]:

```
# Displaying first 5 records
df.head()
```

Out[7]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	

In [8]:

```
# Displaying the statistical values of the data
df.describe()
```

Out[8]:

	age	fnlwgt	education_num	capital_gain	capital_loss	hours_per_week	Income
count	48842.000000	4.884200e+04	48842.000000	48842.000000	48842.000000	48842.000000	48842.000000
mean	38.643585	1.896641e+05	10.078089	1079.067626	87.502314	40.422382	0.160538
std	13.710510	1.056040e+05	2.570973	7452.019058	403.004552	12.391444	0.367108
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000	0.000000
25%	28.000000	1.175505e+05	9.000000	0.000000	0.000000	40.000000	0.000000
50%	37.000000	1.781445e+05	10.000000	0.000000	0.000000	40.000000	0.000000
75%	48.000000	2.376420e+05	12.000000	0.000000	0.000000	45.000000	0.000000
max	90.000000	1.490400e+06	16.000000	99999.000000	4356.000000	99.000000	1.000000

In [9]:

```
# Replacing ? with Nan values
df.replace(' ?', np.nan, inplace=True)
```

In [10]:

```
# Getting the count of Nan/Null values
df.isnull().sum()
```

Out[10]:

```
age                  0
workclass         2799
fnlwgt               0
education            0
education_num        0
marital_status       0
occupation        2809
relationship         0
race                 0
sex                  0
capital_gain         0
capital_loss         0
hours_per_week       0
native_country     857
Income               0
dtype: int64
```

In [11]:

```
# Filling na values with 0
df.fillna(' 0', inplace=True)
```

In [12]:

```
col_in_category = (
    'workclass',
    'education',
    'marital_status',
    'occupation',
    'relationship',
    'race',
    'sex',
    'native_country'
)

from sklearn.preprocessing import LabelEncoder

for col in col_in_category:
    encoder=LabelEncoder()
    df[col]=encoder.fit_transform(df[col])
```

In [13]:

```
df.head()
```

Out[13]:

	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_wee
0	39	7	77516	9	13	4	1	1	4	1	2174	0	4
1	50	6	83311	9	13	2	4	0	4	1	0	0	1
2	38	4	215646	11	9	0	6	1	4	1	0	0	4
3	53	4	234721	1	7	2	6	0	2	1	0	0	4
4	28	4	338409	9	13	2	10	5	2	0	0	0	4

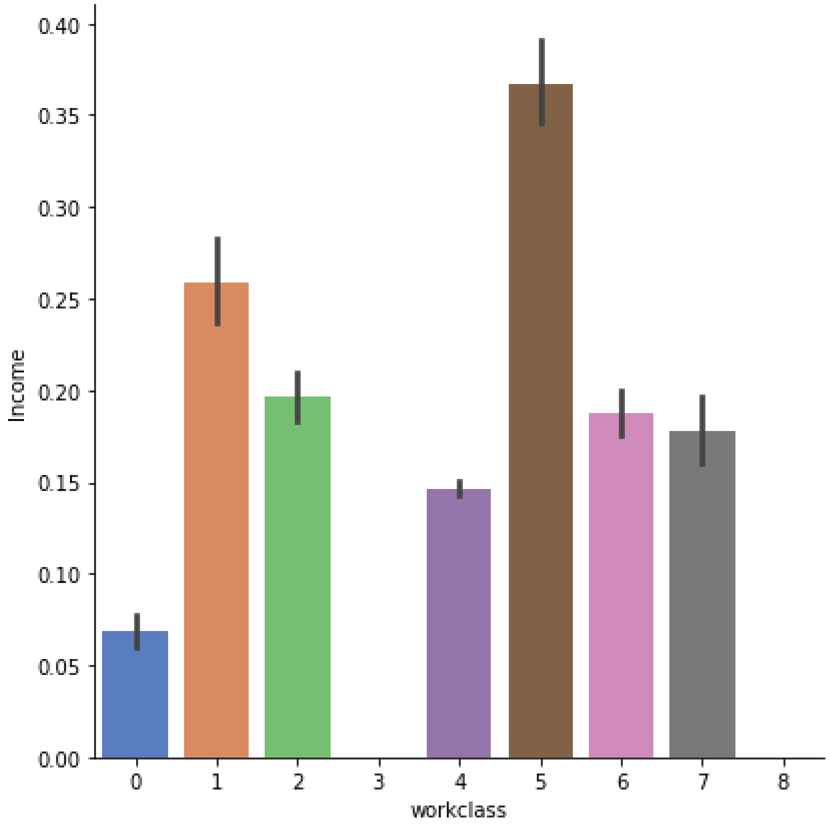
In [14]:

```
# Creating Factor Plot for work and Income features
sns.factorplot(x="workclass", y="Income", data=df, kind="bar", size = 6,
palette = "muted")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3666: UserWarning: The `factorplot` function has bee
n renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the
default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3672: UserWarning: The `size` paramter has been rena
med to `height`; please update your code.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for mu
ltidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpr
eted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[14]:

```
<seaborn.axisgrid.FacetGrid at 0x1c72e761320>
```


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Income
0
1
2
3
4
5
6
7
8
workclass

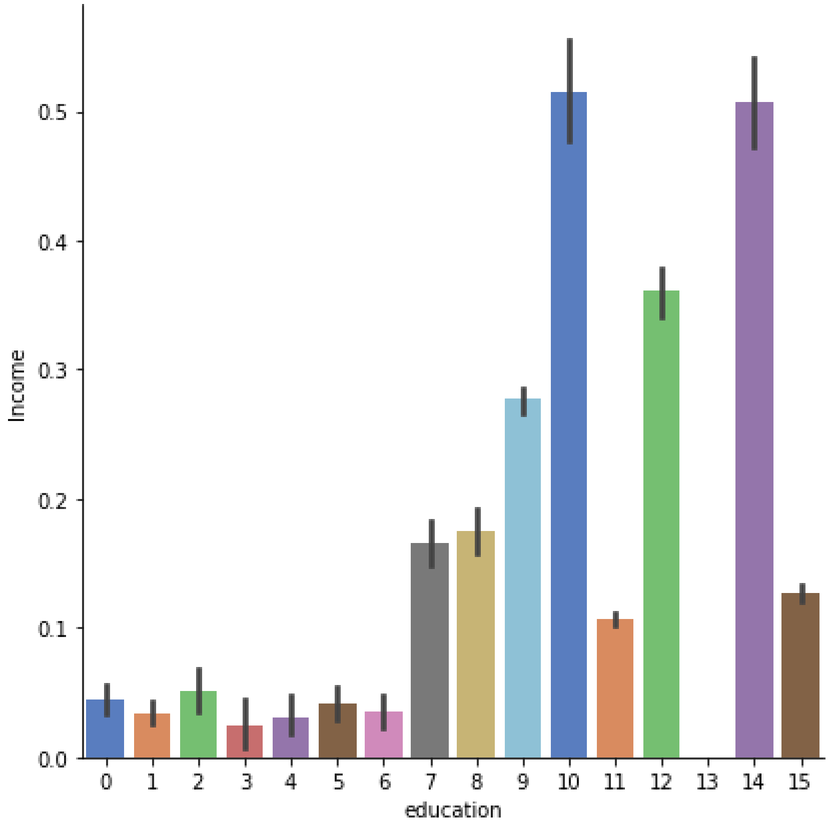
In [15]:

```python
# Creating Factor Plot for Education and Income features
sns.factorplot(x="education",y="Income",data=df,kind="bar", size = 6,
palette = "muted")
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3666: UserWarning: The `factorplot` function has bee
n renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the
default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:3672: UserWarning: The `size` paramter has been rena
med to `height`; please update your code.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for mu
ltidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpr
eted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[15]:

```
<seaborn.axisgrid.FacetGrid at 0x1c72e4ff898>
```



In [16]:
```
from sklearn.model_selection import train_test_split
```

In [17]:
```
# Creating target and Independent variable for algorithm
Y=np.array(df['Income'])
X=np.array(df.drop(['Income'],1))
```

In [18]:
```
# Splitting train and test data
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

In [19]:
```
# Importing XG Boost classifier
import xgboost as xgb
model = xgb.XGBClassifier()
```

In [20]:

```
from sklearn.metrics import accuracy_score
from xgboost.sklearn import XGBClassifier
from sklearn.model_selection import GridSearchCV
learning_rate = [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]
param_grid = dict(learning_rate=learning_rate)
grid_search = GridSearchCV(model, param_grid, scoring="neg_log_loss", n_jobs=-1, cv=10)
```

In [21]:

```
# Learning the training data and fitting them
grid_search.fit(X_train, y_train)
```

Out[21]:

```
GridSearchCV(cv=10, error_score='raise',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1),
       fit_params=None, iid=True, n_jobs=-1,
       param_grid={'learning_rate': [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='neg_log_loss', verbose=0)
```

In [22]:

```
top_param=grid_search.best_estimator_
```

In [23]:

```
best_model=XGBClassifier(learning_rate=top_param.learning_rate,booster=top_param.booster,gamma=top_param.gamma,n_estimator
```

In [24]:

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

Out[24]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.3, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [25]:

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

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth value of
an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to chec
k that an array is not empty.
  if diff:
```

Out[25]:

```
0.8716505003455072
```

In [26]:

```
# fitting againist test data
best_model.score(X_test,y_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\label.py:151: DeprecationWarning: The truth value of
an empty array is ambiguous. Returning False, but in future this will result in an error. Use `array.size > 0` to chec
k that an array is not empty.
  if diff:
```

Out[26]:

```
0.8604770191421844
```

In [ ]:

```

```