In [1]:
```
#Employee Salary prediction using adult csv
#load your library

import pandas as pd
```

In [2]:
```
data=pd.read_csv('adult.csv')
```

In [3]:
```
data
```

Out[3]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-chil
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husban
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husban
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husban
4	18	?	103497	Some-college	10	Never-married	?	Own-chil
...	...	...	...	...	...	...	...	.
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wif
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husban
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-chil
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wif

48842 rows × 15 columns

In [4]:
```
data.shape
```

Out[4]: (48842, 15)

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

Out[5]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	ra
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Bl
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	Wh
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	Wh
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Bl
4	18	?	103497	Some-college	10	Never-married	?	Own-child	Wh

In [6]:
```
data.head(7)
```

Out[6]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	ra
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Bl
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	Wh
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	Wh
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Bl
4	18	?	103497	Some-college	10	Never-married	?	Own-child	Wh
5	34	Private	198693	10th	6	Never-married	Other-service	Not-in-family	Wh
6	29	?	227026	HS-grad	9	Never-married	?	Unmarried	Bl

In [7]:
```
data.tail()
```

Out[7]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wif
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husban
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-chil
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wif

In [8]:
```
data.tail(7)
```

Out[8]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
48835	53	Private	321865	Masters	14	Married-civ-spouse	Exec-managerial	Husban
48836	22	Private	310152	Some-college	10	Never-married	Protective-serv	Not-in famil
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wif
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husban
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-chil
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wif

In [9]:
```
#NULL VALUES
data.isna()
```

Out[9]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	..
48837	False	False	False	False	False	False	False	False
48838	False	False	False	False	False	False	False	False
48839	False	False	False	False	False	False	False	False
48840	False	False	False	False	False	False	False	False
48841	False	False	False	False	False	False	False	False

48842 rows × 15 columns

In [10]:
```
#null values
data.isna().sum() #mean mdeian mode arbitrary
```

Out[10]:
```
age                0
workclass          0
fnlwgt             0
education          0
educational-num    0
marital-status     0
occupation         0
relationship       0
race               0
gender             0
capital-gain       0
capital-loss       0
hours-per-week     0
native-country     0
income             0
dtype: int64
```

In [11]:
```
print(data.occupation.value_counts())
```

```
occupation
Prof-specialty       6172
Craft-repair         6112
Exec-managerial      6086
Adm-clerical         5611
Sales                5504
Other-service        4923
Machine-op-inspct    3022
?                    2809
Transport-moving     2355
Handlers-cleaners    2072
Farming-fishing      1490
Tech-support         1446
Protective-serv       983
Priv-house-serv       242
Armed-Forces           15
Name: count, dtype: int64
```

In [12]:
```python
print(data.gender.value_counts())
```

```
gender
Male      32650
Female    16192
Name: count, dtype: int64
```

In [13]:
```python
print(data.education.value_counts())
```

```
education
HS-grad         15784
Some-college    10878
Bachelors        8025
Masters          2657
Assoc-voc        2061
11th             1812
Assoc-acdm       1601
10th             1389
7th-8th           955
Prof-school       834
9th               756
12th              657
Doctorate         594
5th-6th           509
1st-4th           247
Preschool          83
Name: count, dtype: int64
```

In [14]:
```python
print(data['marital-status'].value_counts())
```

```
marital-status
Married-civ-spouse       22379
Never-married            16117
Divorced                  6633
Separated                 1530
Widowed                   1518
Married-spouse-absent      628
Married-AF-spouse           37
Name: count, dtype: int64
```

In [15]:
```
print(data.workclass.value_counts())
```

```
workclass
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
?                    2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Without-pay            21
Never-worked           10
Name: count, dtype: int64
```

In [16]:
```
print(data.relationship.value_counts())
```

```
relationship
Husband           19716
Not-in-family     12583
Own-child          7581
Unmarried          5125
Wife               2331
Other-relative     1506
Name: count, dtype: int64
```

In [17]:
```
print(data.race.value_counts())
```

```
race
White                 41762
Black                  4685
Asian-Pac-Islander     1519
Amer-Indian-Eskimo      470
Other                   406
Name: count, dtype: int64
```

In [19]:
```
data.workclass.replace({'?':'Others'},inplace=True)
```

In [20]:
```
print(data.occupation.value_counts())
```

```
occupation
Prof-specialty       6172
Craft-repair         6112
Exec-managerial      6086
Adm-clerical         5611
Sales                5504
Other-service        4923
Machine-op-inspct    3022
?                    2809
Transport-moving     2355
Handlers-cleaners    2072
Farming-fishing      1490
Tech-support         1446
Protective-serv       983
Priv-house-serv       242
Armed-Forces           15
Name: count, dtype: int64
```

In [21]:
```
data
```

Out[21]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationshi
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-chil
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husban
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husban
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husban
4	18	Others	103497	Some-college	10	Never-married	?	Own-chil
...	...	...	...	...	...	...	...	.
48837	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wif
48838	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husban
48839	58	Private	151910	HS-grad	9	Widowed	Adm-clerical	Unmarrie
48840	22	Private	201490	HS-grad	9	Never-married	Adm-clerical	Own-chil
48841	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial	Wif

48842 rows × 15 columns

In [22]:
```
data.workclass.replace({'?':'notlisted'},inplace=True)
```

In [23]:
```
print(data.workclass.value_counts())
```

```
workclass
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
Others               2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Without-pay            21
Never-worked           10
Name: count, dtype: int64
```

In [24]:
```
data=data[data['workclass']!='Without-pay']
data=data[data['workclass']!='Never-worked']
```

In [25]:
```
print(data['workclass'].value_counts())
```

```
workclass
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
Others               2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Name: count, dtype: int64
```

In [26]:
```
data.shape
```

Out[26]:
```
(48811, 15)
```

In [27]:
```
data=data[data['education']!='5th-6th']
data=data[data['education']!='1st-4th']
data=data[data['education']!='Preschool']
```

In [28]:
```
print(data.education.value_counts())
```

```
education
HS-grad         15768
Some-college    10873
Bachelors        8025
Masters          2657
Assoc-voc        2061
11th             1809
Assoc-acdm       1599
10th             1387
7th-8th           952
Prof-school       834
9th               756
12th              657
Doctorate         594
Name: count, dtype: int64
```

In [29]:
```
data.shape
```

Out[29]: (47972, 15)

In [30]:
```
data=data[data['relationship']!='Wife']
data=data[data['relationship']!='Other-relative']
```

In [31]:
```
print(data.relationship.value_counts())
```

```
relationship
Husband          19336
Not-in-family    12372
Own-child         7529
Unmarried         5023
Name: count, dtype: int64
```

In [32]:
```
data.shape
```

Out[32]: (44260, 15)

In [33]:
```
data=data[data['race']!='Amer-Indian-Eskimo']
data=data[data['race']!='Other']
```

In [34]:
```
print(data.race.value_counts())
```

```
race
White                 38136
Black                  4145
Asian-Pac-Islander     1258
Name: count, dtype: int64
```

In [35]:
```
data.shape
```

Out[35]: (43539, 15)

In [36]:
```
data=data.drop(columns=['education'])
```

In [37]:
```
data
```

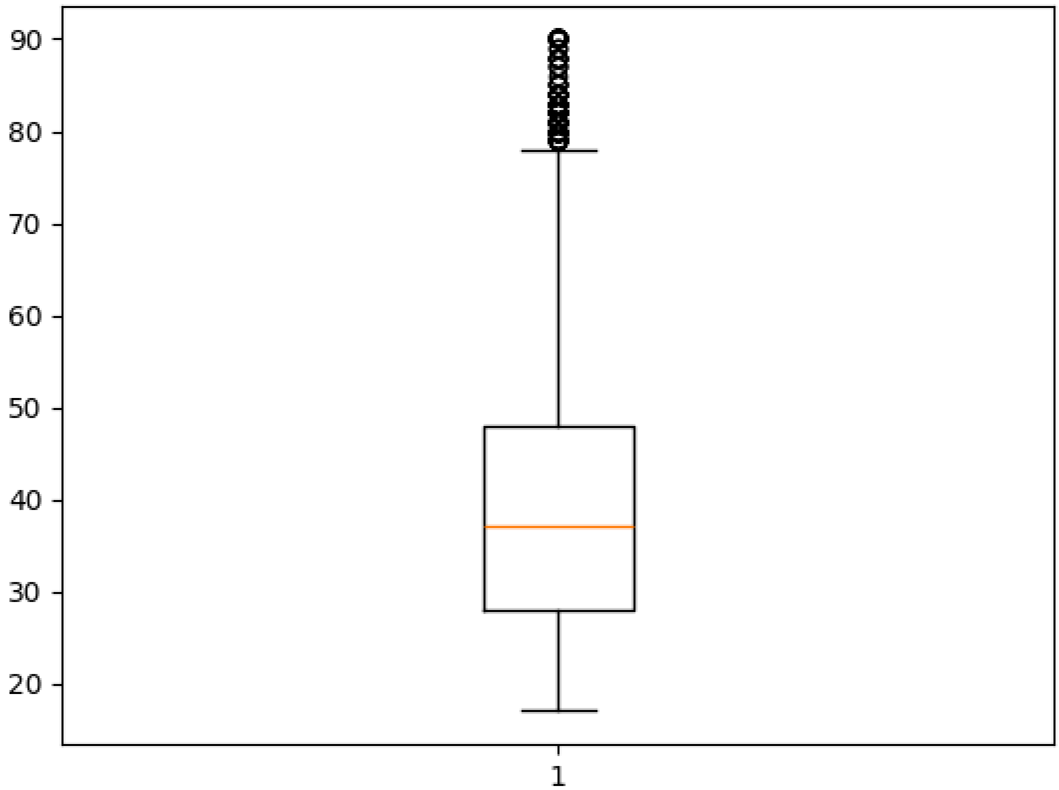
Out[37]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	g
0	25	Private	226802	7	Never-married	Machine-op-inspct	Own-child	Black	
1	38	Private	89814	9	Married-civ-spouse	Farming-fishing	Husband	White	
2	28	Local-gov	336951	12	Married-civ-spouse	Protective-serv	Husband	White	
3	44	Private	160323	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	
4	18	Others	103497	10	Never-married	?	Own-child	White	F
...	...	...	...	...	...	...	...	...	
48835	53	Private	321865	14	Married-civ-spouse	Exec-managerial	Husband	White	
48836	22	Private	310152	10	Never-married	Protective-serv	Not-in-family	White	
48838	40	Private	154374	9	Married-civ-spouse	Machine-op-inspct	Husband	White	
48839	58	Private	151910	9	Widowed	Adm-clerical	Unmarried	White	F
48840	22	Private	201490	9	Never-married	Adm-clerical	Own-child	White	

43539 rows × 14 columns

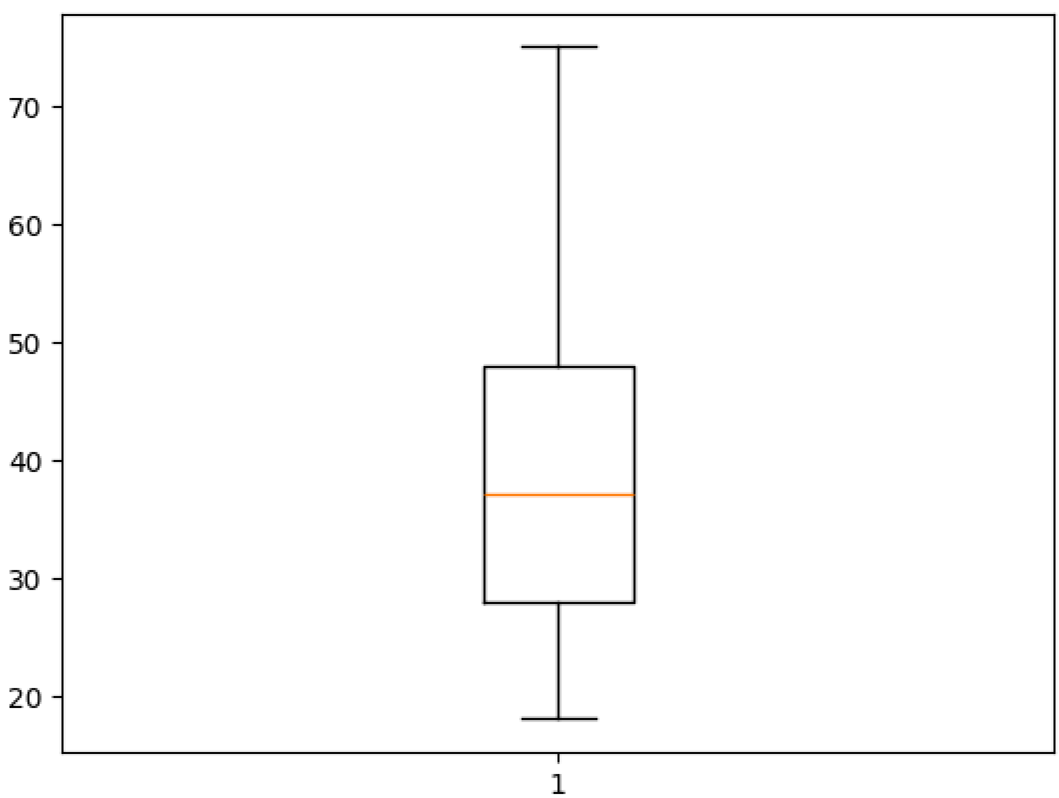
In [39]:

```
#outlier ( #others column (marital-status       occupation       relationship))
import matplotlib.pyplot as plt
plt.boxplot(data['age'])
plt.show()
```



In [40]:
```
data=data[(data['age']<=75)& (data['age']>17)]
```

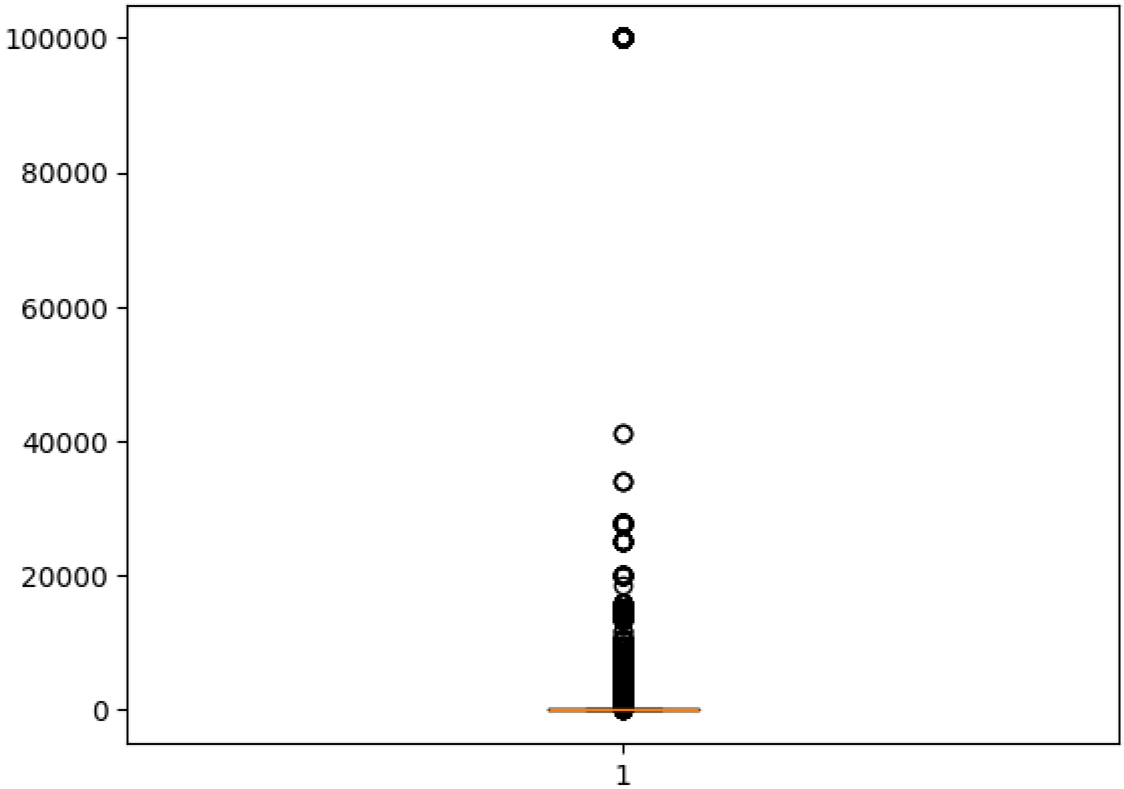
In [41]:
```
plt.boxplot(data['age'])
plt.show()
```



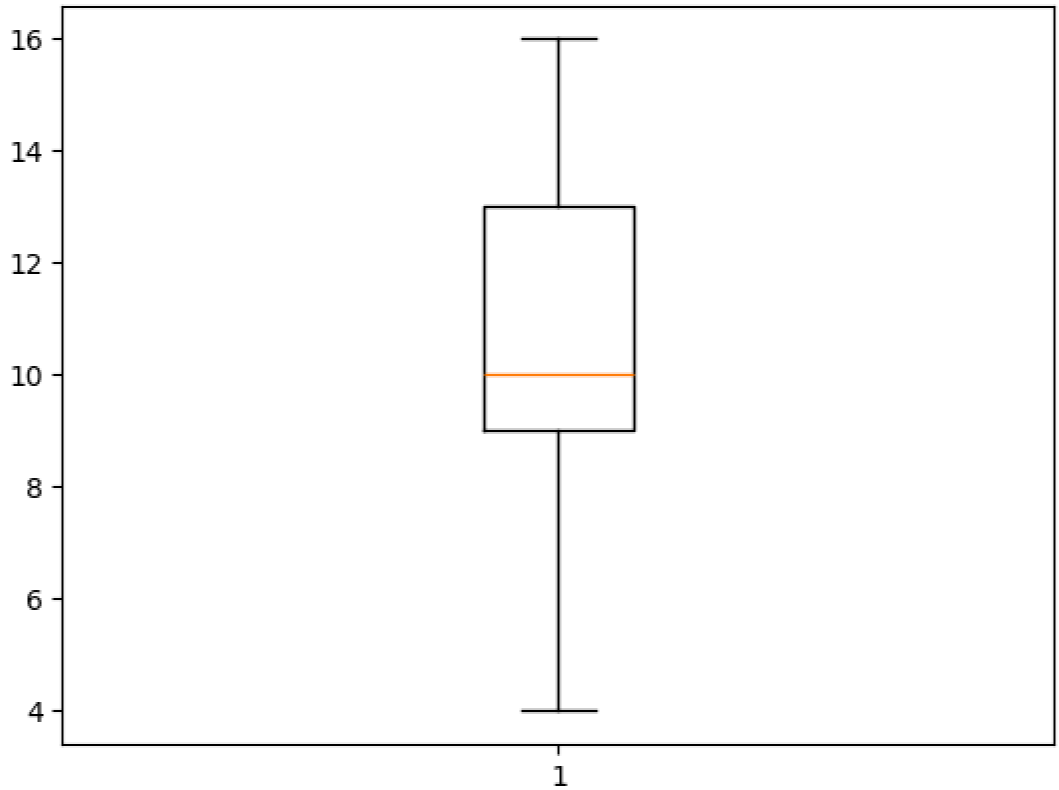
In [42]:
```
data.shape
```

Out[42]:
```
(42652, 14)
```

In [43]:
```
plt.boxplot(data['capital-gain'])
plt.show()
```

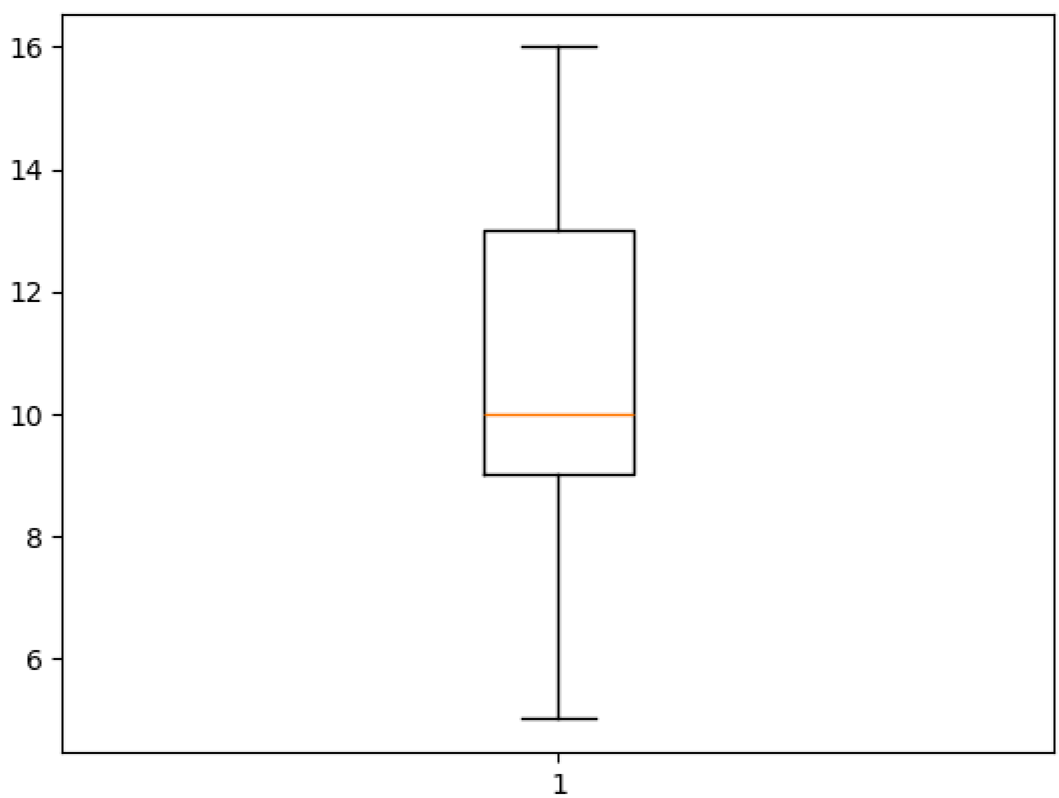


In [44]:
```
plt.boxplot(data['educational-num'])
plt.show()
```

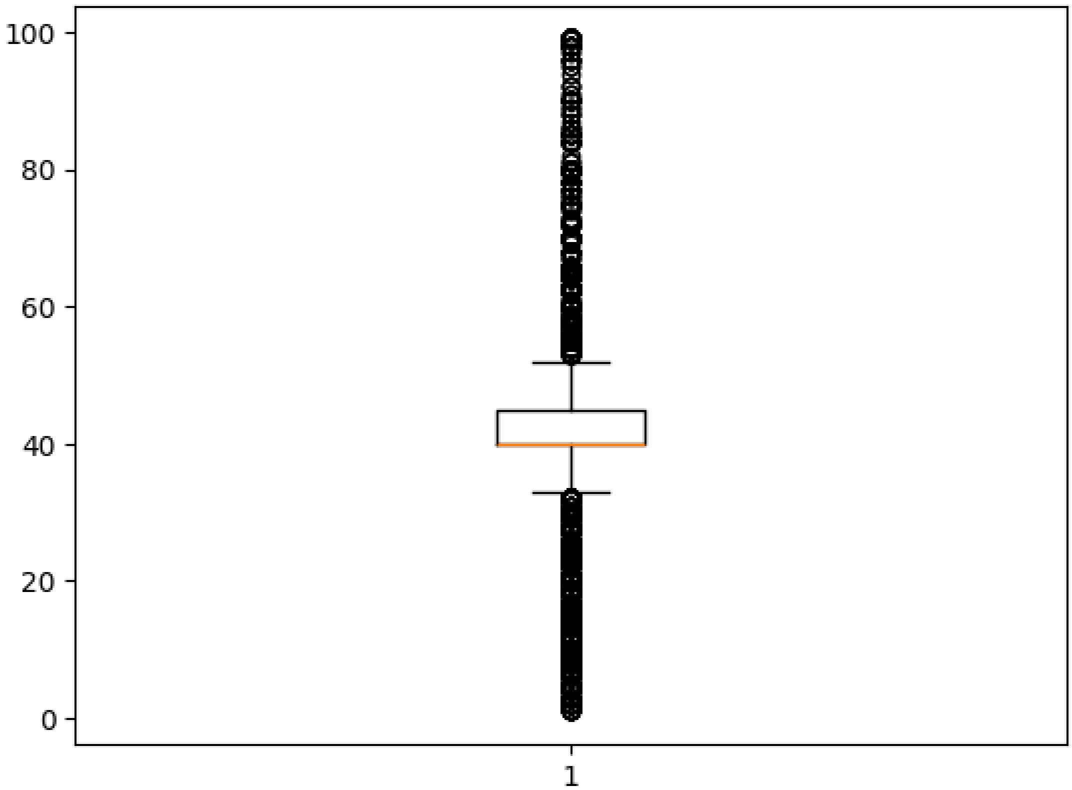


In [45]:
```
data=data[(data['educational-num']<=16)&(data['educational-num']>=5)]
```

In [46]:
```
plt.boxplot(data['educational-num'])
plt.show()
```



In [47]:
```
plt.boxplot(data['hours-per-week'])
plt.show()
```



In [48]:
```
data.shape
```

Out[48]: (41860, 14)

In [49]:
```
#label encodin
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data['workclass']=encoder.fit_transform(data['workclass'])
data['marital-status']=encoder.fit_transform(data['marital-status'])
data['occupation']=encoder.fit_transform(data['occupation'])
data['relationship']=encoder.fit_transform(data['relationship'])
data['race']=encoder.fit_transform(data['race'])
data['gender']=encoder.fit_transform(data['gender'])
data['native-country']=encoder.fit_transform(data['native-country'])
data
```

Out[49]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gen
0	25	3	226802	7	4	7	2	1	
1	38	3	89814	9	2	5	0	2	
2	28	1	336951	12	2	11	0	2	
3	44	3	160323	10	2	7	0	1	
4	18	2	103497	10	4	0	2	2	
...	...	...	...	...	...	...	...	...	
48835	53	3	321865	14	2	4	0	2	
48836	22	3	310152	10	4	11	1	2	
48838	40	3	154374	9	2	7	0	2	
48839	58	3	151910	9	6	1	3	2	
48840	22	3	201490	9	4	1	2	2	

41860 rows × 14 columns

In [50]:

```
x=data.drop(columns=['income']) #input data
y=data['income'] #output data
x
```

Out[50]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gen
0	25	3	226802	7	4	7	2	1	
1	38	3	89814	9	2	5	0	2	
2	28	1	336951	12	2	11	0	2	
3	44	3	160323	10	2	7	0	1	
4	18	2	103497	10	4	0	2	2	
...	...	...	...	...	...	...	...	...	
48835	53	3	321865	14	2	4	0	2	
48836	22	3	310152	10	4	11	1	2	
48838	40	3	154374	9	2	7	0	2	
48839	58	3	151910	9	6	1	3	2	
48840	22	3	201490	9	4	1	2	2	

41860 rows × 13 columns

In [51]:

```
x=data.drop(columns=['income'])
y=data['income']
```

In [52]:

```
y
```

Out[52]:

```
0         <=50K
1         <=50K
2          >50K
3          >50K
4         <=50K
          ...
48835      >50K
48836     <=50K
48838      >50K
48839     <=50K
48840     <=50K
Name: income, Length: 41860, dtype: object
```

In [53]:

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x=scaler.fit_transform(x)
x
```

Out[53]:
```
array([[0.12280702, 0.5       , 0.14426962, ..., 0.        , 0.39795918,
        0.95      ],
       [0.35087719, 0.5       , 0.05149899, ..., 0.        , 0.5       ,
        0.95      ],
       [0.1754386 , 0.16666667, 0.21886443, ..., 0.        , 0.39795918,
        0.95      ],
       ...,
       [0.38596491, 0.5       , 0.09522013, ..., 0.        , 0.39795918,
        0.95      ],
       [0.70175439, 0.5       , 0.09355147, ..., 0.        , 0.39795918,
        0.95      ],
       [0.07017544, 0.5       , 0.1271279 , ..., 0.        , 0.19387755,
        0.95      ]], shape=(41860, 13))
```

In [54]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest= train_test_split(x,y, test_size=0.2, random_state=23,
```

In [55]:
```
xtrain
```

Out[55]:
```
array([[0.45614035, 0.5       , 0.16856953, ..., 0.34527089, 0.44897959,
        0.95      ],
       [0.12280702, 0.5       , 0.14996909, ..., 0.        , 0.39795918,
        0.95      ],
       [0.40350877, 0.5       , 0.13973701, ..., 0.        , 0.55102041,
        0.95      ],
       ...,
       [0.64912281, 0.5       , 0.25734188, ..., 0.        , 0.39795918,
        0.95      ],
       [0.22807018, 0.5       , 0.06656572, ..., 0.        , 0.39795918,
        0.95      ],
       [0.1754386 , 0.16666667, 0.10733961, ..., 0.        , 0.44897959,
        0.95      ]], shape=(33488, 13))
```

In [56]:
```
#machine learning algorithm
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(xtrain, ytrain) #input and output training data
predict=knn.predict(xtest)
predict #predict value
```

Out[56]:
```
array(['<=50K', '>50K', '<=50K', ..., '>50K', '>50K', '<=50K'],
      shape=(8372,), dtype=object)
```

In [57]:
```
from sklearn.metrics import accuracy_score
accuracy_score(ytest, predict)
```

Out[57]:
```
0.8190396559961778
```

In [58]:
```
#Deep learning Algorithm
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(xtrain, ytrain) #input and output training data
predict1=lr.predict(xtest)
predict1 #predict value
```

```
Out[58]: array(['<=50K', '<=50K', '<=50K', ..., '>50K', '>50K', '<=50K'],
               shape=(8372,), dtype=object)
```

```
In [59]: from sklearn.metrics import accuracy_score
         accuracy_score(ytest, predict1)
```

```
Out[59]: 0.8324175824175825
```

```
In [60]: from sklearn.neural_network import MLPClassifier
         clf=MLPClassifier(solver='adam', hidden_layer_sizes=(5,2), random_state=2, max_iter
         clf.fit(xtrain, ytrain)
         predict2=clf.predict(xtest)
         predict2
```

```
Out[60]: array(['<=50K', '<=50K', '<=50K', ..., '>50K', '>50K', '<=50K'],
               shape=(8372,), dtype='<U5')
```

```
In [61]: from sklearn.metrics import accuracy_score
         accuracy_score(ytest, predict2)
```

```
Out[61]: 0.8338509316770186
```

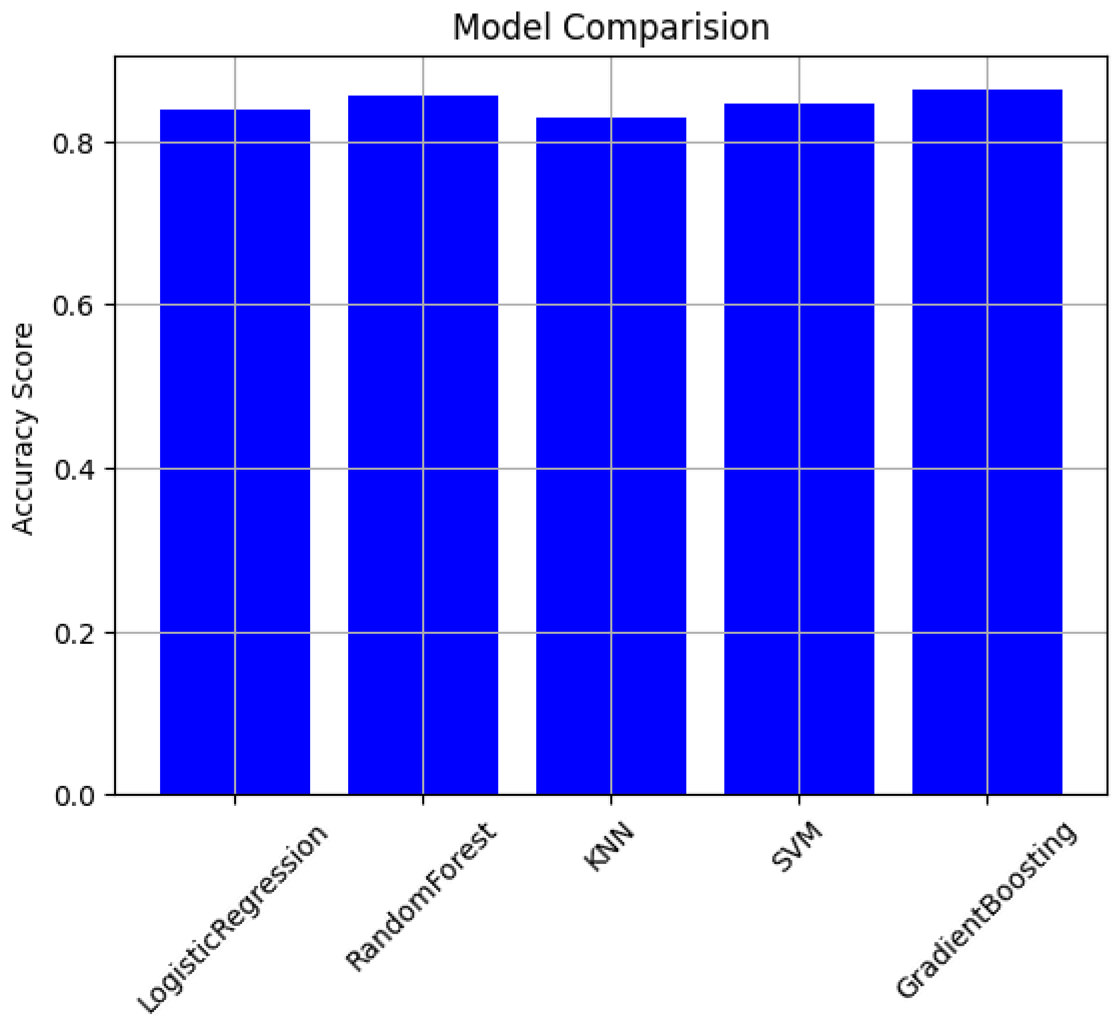
```
In [62]: from sklearn.pipeline import Pipeline
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score, classification_report
         from sklearn.linear_model import LogisticRegression
         from sklearn.ensemble  import RandomForestClassifier, GradientBoostingClassifier
         from sklearn.neighbors  import KNeighborsClassifier
         from sklearn.svm import SVC
         from sklearn.preprocessing import StandardScaler, OneHotEncoder

         X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta

         models = {
             "LogisticRegression":LogisticRegression(),
             "RandomForest": RandomForestClassifier(),
             "KNN": KNeighborsClassifier(),
             "SVM": SVC(),
             "GradientBoosting": GradientBoostingClassifier()
         }

         results = {}

         for name, model in models.items():
             pipe = Pipeline([
                 ('scaler', StandardScaler()),
                 ('model', model)
             ])

             pipe.fit(X_train, y_train)
             y_pred = pipe.predict(X_test)
             acc = accuracy_score(y_test, y_pred)
             results[name] = acc
             print(f"{name} Accuracy: {acc:.4f}")
```

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

```
LogisticRegression Accuracy: 0.8395
              precision    recall  f1-score   support

       <=50K       0.86      0.93      0.90      6263
        >50K       0.74      0.56      0.64      2109

    accuracy                           0.84      8372
   macro avg       0.80      0.75      0.77      8372
weighted avg       0.83      0.84      0.83      8372

RandomForest Accuracy: 0.8548
              precision    recall  f1-score   support

       <=50K       0.88      0.94      0.91      6263
        >50K       0.76      0.61      0.68      2109

    accuracy                           0.85      8372
   macro avg       0.82      0.78      0.79      8372
weighted avg       0.85      0.85      0.85      8372

KNN Accuracy: 0.8280
              precision    recall  f1-score   support

       <=50K       0.87      0.91      0.89      6263
        >50K       0.69      0.58      0.63      2109

    accuracy                           0.83      8372
   macro avg       0.78      0.75      0.76      8372
weighted avg       0.82      0.83      0.82      8372

SVM Accuracy: 0.8465
              precision    recall  f1-score   support

       <=50K       0.86      0.95      0.90      6263
        >50K       0.78      0.55      0.64      2109

    accuracy                           0.85      8372
   macro avg       0.82      0.75      0.77      8372
weighted avg       0.84      0.85      0.84      8372

GradientBoosting Accuracy: 0.8622
              precision    recall  f1-score   support

       <=50K       0.88      0.95      0.91      6263
        >50K       0.80      0.60      0.69      2109

    accuracy                           0.86      8372
   macro avg       0.84      0.78      0.80      8372
weighted avg       0.86      0.86      0.86      8372
```

In [63]:
```python
import matplotlib.pyplot as plt
```

```
plt.bar(results.keys(), results.values(), color='blue')
plt.ylabel('Accuracy Score')
plt.title('Model Comparision')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [64]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import joblib

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_sta

# Define models
models = {
    "LogisticRegression": LogisticRegression(max_iter=1000),
    "RandomForest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
```

```python
    "GradientBoosting": GradientBoostingClassifier()
}

results = {}

# Train and evaluate
for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc = accuracy_score(y_test, preds)
    results[name] = acc
    print(f"{name}: {acc:.4f}")

# Get best model
best_model_name = max(results, key=results.get)
best_model = models[best_model_name]
print(f"\n✅ Best model: {best_model_name} with accuracy {results[best_model_name]

# Save the best model
joblib.dump(best_model, "best_model.pkl")
print("✅ Saved best model as best_model.pkl")
```

```
LogisticRegression: 0.8364
RandomForest: 0.8546
KNN: 0.8208
SVM: 0.8438
GradientBoosting: 0.8622

✅ Best model: GradientBoosting with accuracy 0.8622
✅ Saved best model as best_model.pkl
```