In [1]:
```
import pandas as pd
```

In [2]:
```
data=pd.read_csv(r"C:\Users\lavan\Downloads\adult 3.csv")
```

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

Out[3]:

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

48842 rows × 15 columns

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

Out[4]:
```
(48842, 15)
```

In [5]:
```
data.head(7)
```

Out[5]:

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

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

Out[6]:

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

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

Out[7]:

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

48842 rows × 15 columns

In [8]:
```
data.isna().sum()
```

Out[8]:
```
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 [9]:
```
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 [10]:
```
print(data.gender.value_counts())
```

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

In [11]:
```
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 [12]:
```
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 [13]:
```
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 [14]:
```
data.occupation.replace({'?':'Others'},inplace=True)
```

In [15]:
```
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
Others               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 [16]:
```
data
```

Out[16]:

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

48842 rows × 15 columns

In [17]:
```
data.workclass.replace({'?':'NotListed'},inplace=True)
```

In [18]:
```
print(data['workclass'].value_counts())
```

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

In [19]:
```
data=data[data['workclass']!='without-pay']
data=data[data['workclass']!='Never-worked']
```

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

```
workclass
Private             33906
Self-emp-not-inc     3862
Local-gov            3136
NotListed            2799
State-gov            1981
Self-emp-inc         1695
Federal-gov          1432
Without-pay            21
Name: count, dtype: int64
```

In [21]:
```
data.shape
```

Out[21]:
```
(48832, 15)
```

In [22]:
```
data=data[data['education']!='5th-6th']
data=data[data['education']!='1st-4th']
data=data[data['education']!='preschool']
```

In [23]:
```
print(data['educational-num'].value_counts())
```

```
educational-num
9     15782
10    10876
13     8025
14     2657
11     2061
7      1809
12     1601
6      1387
4       954
15      834
5       756
8       657
16      594
1        83
Name: count, dtype: int64
```

In [24]:
```
data.shape
```

Out[24]:
```
(48076, 15)
```

In [25]:
```
data.drop(columns=['education'],inplace=True)
```

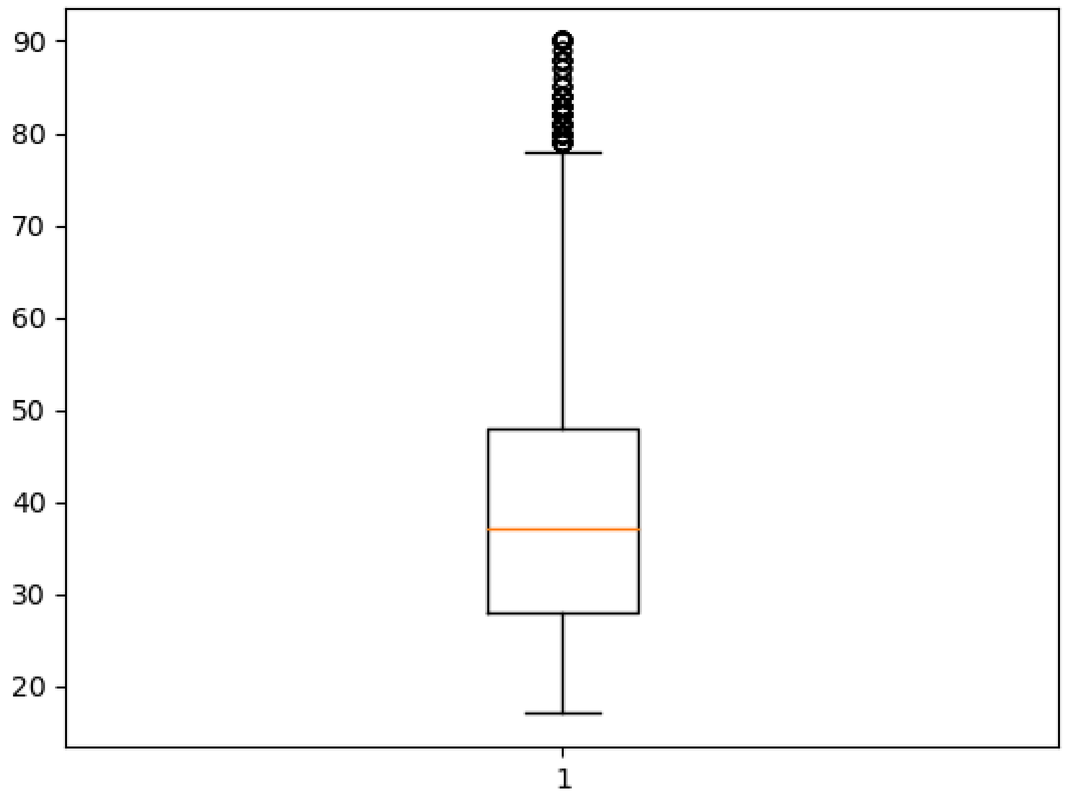
In [26]:
```
data
```

Out[26]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gender
0	25	Private	226802	7	Never-married	Machine-op-inspct	Own-child	Black	Male
1	38	Private	89814	9	Married-civ-spouse	Farming-fishing	Husband	White	Male
2	28	Local-gov	336951	12	Married-civ-spouse	Protective-serv	Husband	White	Male
3	44	Private	160323	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male
4	18	NotListed	103497	10	Never-married	Others	Own-child	White	Female
...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	12	Married-civ-spouse	Tech-support	Wife	White	Female
48838	40	Private	154374	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male
48839	58	Private	151910	9	Widowed	Adm-clerical	Unmarried	White	Female
48840	22	Private	201490	9	Never-married	Adm-clerical	Own-child	White	Male
48841	52	Self-emp-inc	287927	9	Married-civ-spouse	Exec-managerial	Wife	White	Female

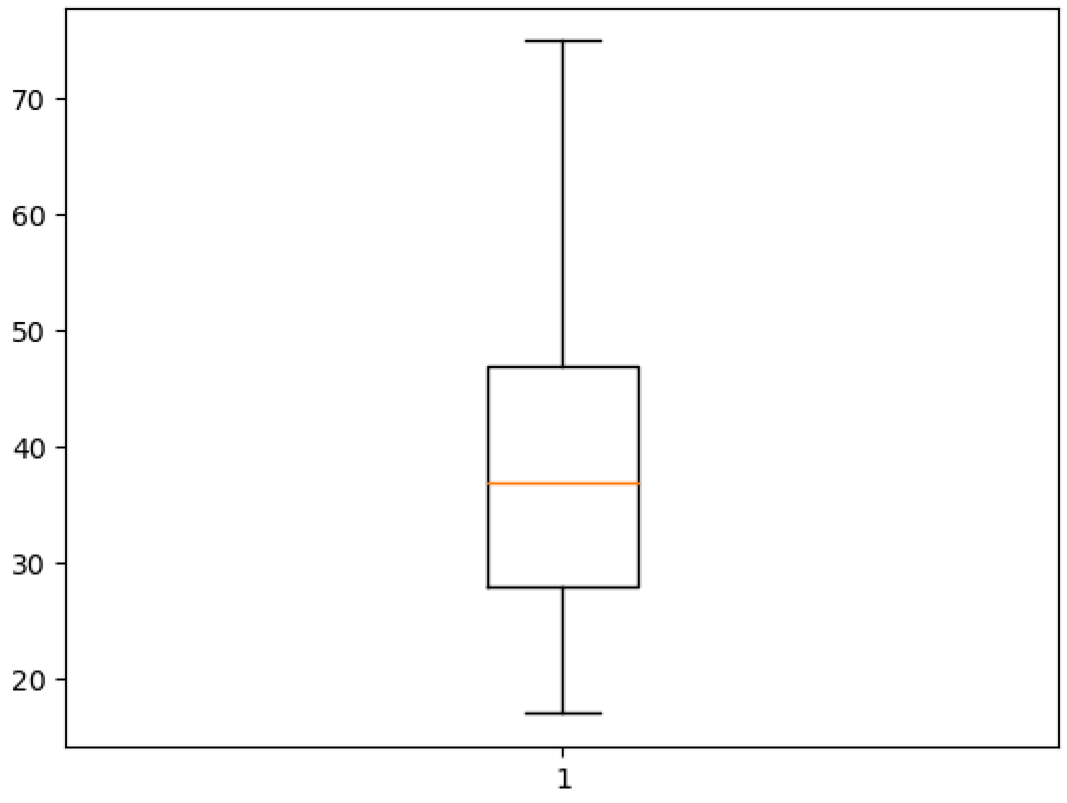
48076 rows × 14 columns

In [27]:
```
import matplotlib.pyplot as plt
plt.boxplot(data['age'])
plt.show()
```



In [28]:
```
data= data[(data['age']<=75) & (data['age']>=17)]
```

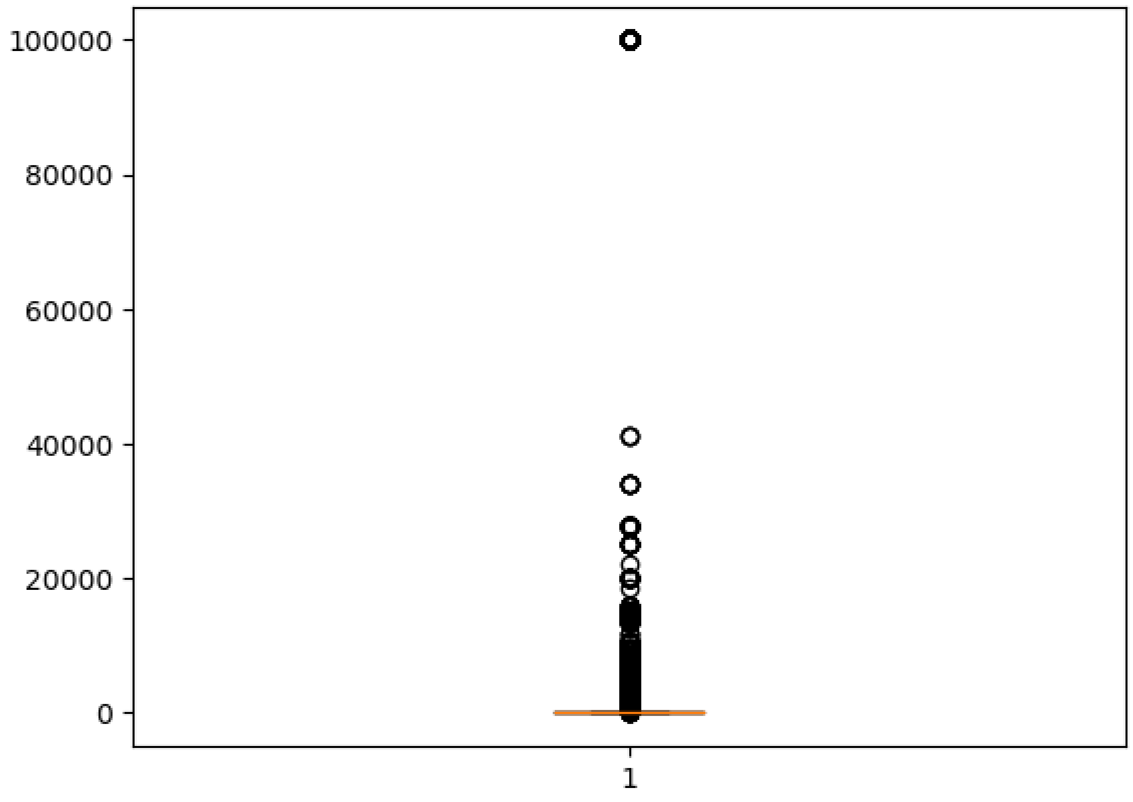
In [29]:
```
plt.boxplot(data['age'])
plt.show()
```



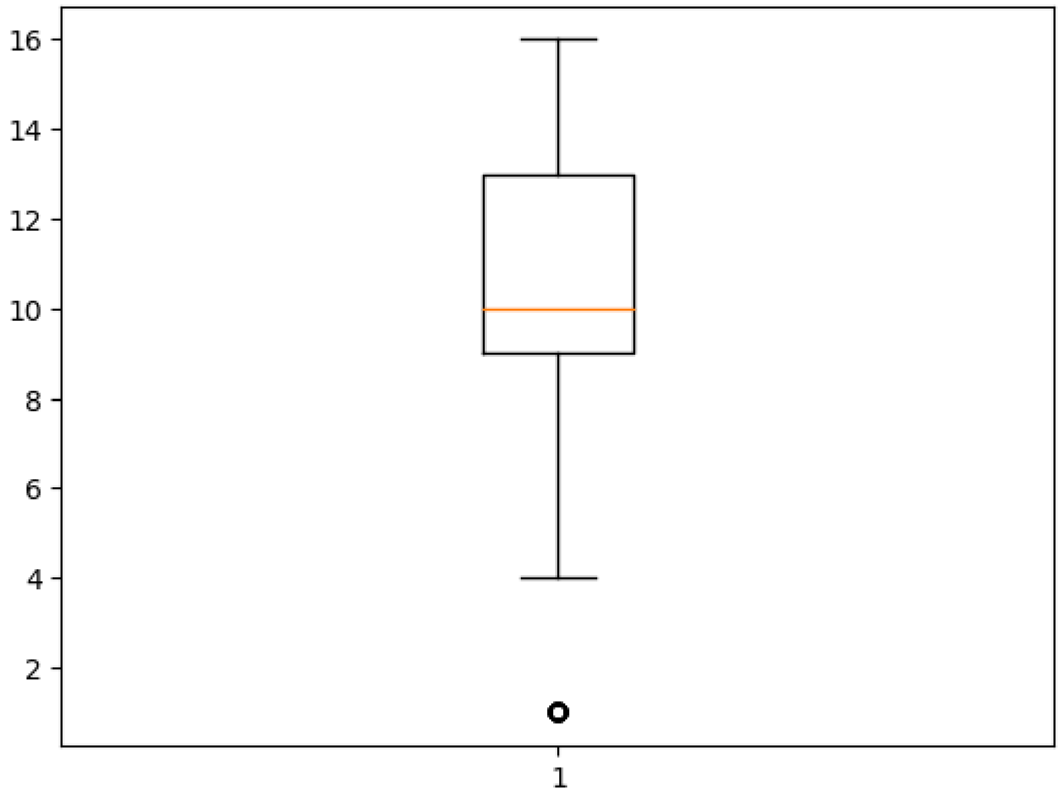
In [30]:
```
data.shape
```

Out[30]:
```
(47722, 14)
```

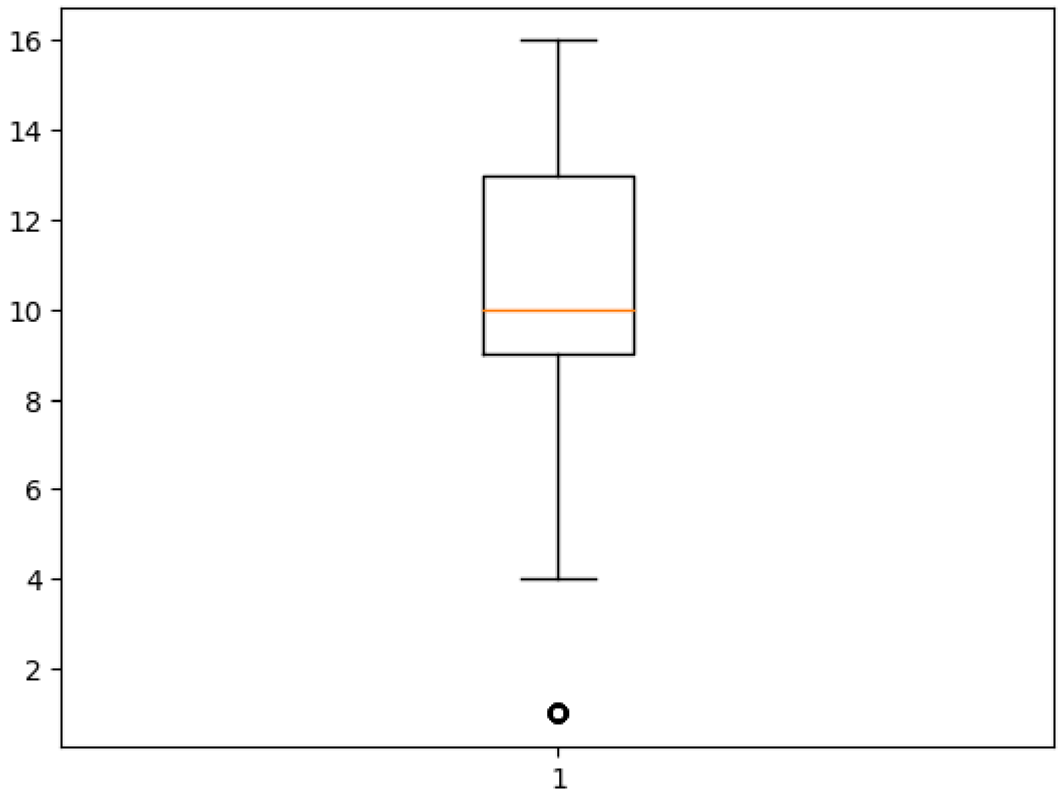
In [31]:
```
plt.boxplot(data['capital-gain'])
plt.show()
```



```
In [32]: plt.boxplot(data['educational-num'])
         plt.show()
```

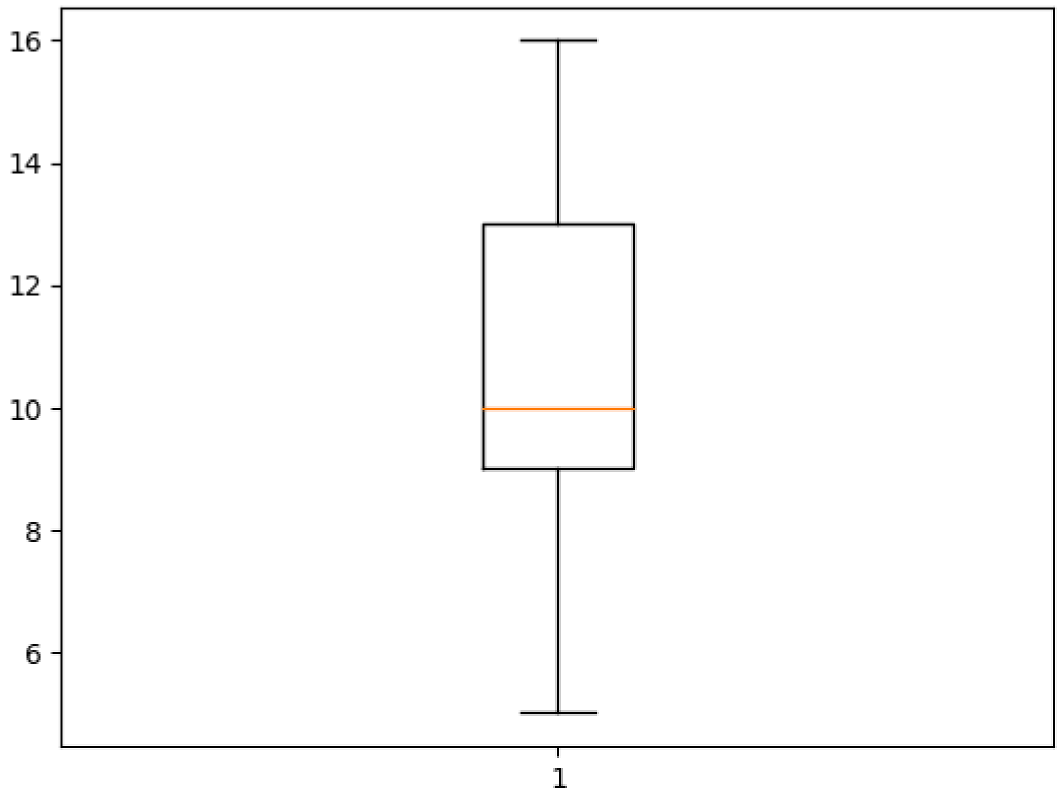


```
In [33]: plt.boxplot(data['educational-num'])
         plt.show()
```

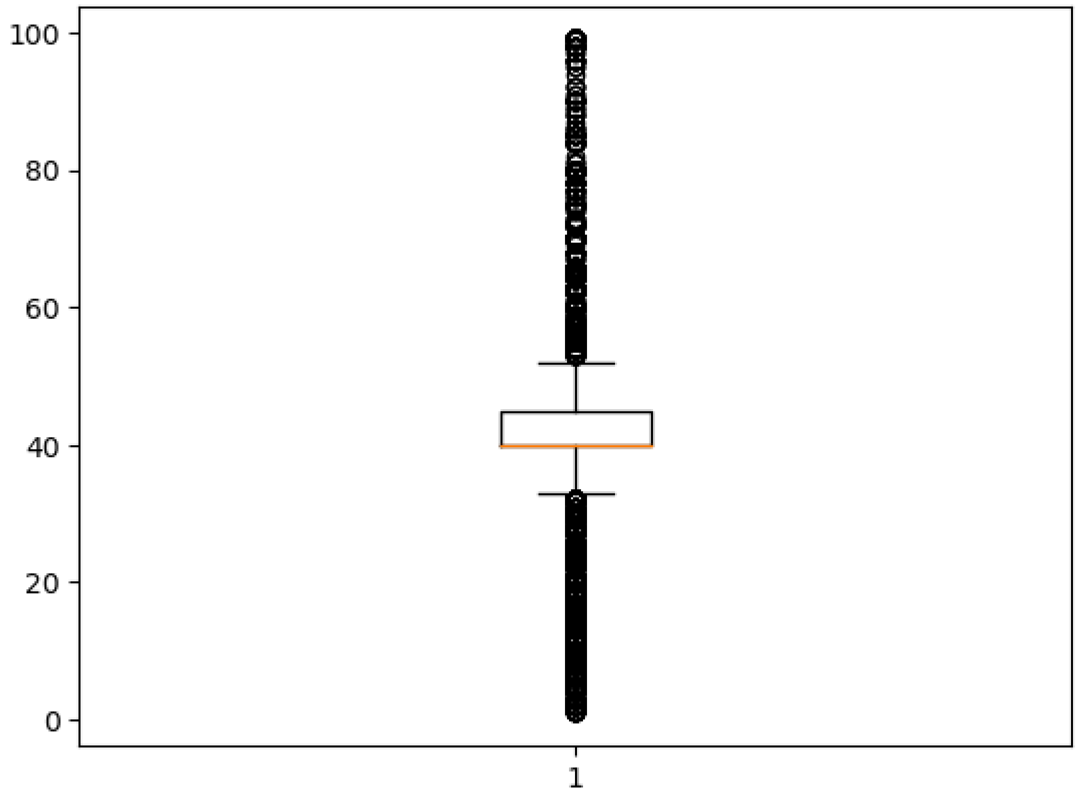


```
In [34]: data=data[(data['educational-num']<=16)&(data['educational-num']>=5)]
```

```
In [35]: plt.boxplot(data['educational-num'])
         plt.show()
```



```
In [36]: plt.boxplot(data['hours-per-week'])
         plt.show()
```



In [37]:
```
data.shape
```

Out[37]:
```
(46739, 14)
```

In [38]:
```
data
```

Out[38]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gender
0	25	Private	226802	7	Never-married	Machine-op-inspct	Own-child	Black	Male
1	38	Private	89814	9	Married-civ-spouse	Farming-fishing	Husband	White	Male
2	28	Local-gov	336951	12	Married-civ-spouse	Protective-serv	Husband	White	Male
3	44	Private	160323	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male
4	18	NotListed	103497	10	Never-married	Others	Own-child	White	Female
...	...	...	...	...	...	...	...	...	...
48837	27	Private	257302	12	Married-civ-spouse	Tech-support	Wife	White	Female
48838	40	Private	154374	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male
48839	58	Private	151910	9	Widowed	Adm-clerical	Unmarried	White	Female
48840	22	Private	201490	9	Never-married	Adm-clerical	Own-child	White	Male
48841	52	Self-emp-inc	287927	9	Married-civ-spouse	Exec-managerial	Wife	White	Female

46739 rows × 14 columns

In [39]:
```
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'])
```

In [40]:
```
data
```

Out[40]:

	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gender	ca
0	25	3	226802	7	4	6	3	2	1	
1	38	3	89814	9	2	4	0	4	1	
2	28	1	336951	12	2	11	0	4	1	
3	44	3	160323	10	2	6	0	2	1	
4	18	2	103497	10	4	8	3	4	0	
...	...	...	...	...	...	...	...	...	...	
48837	27	3	257302	12	2	13	5	4	0	
48838	40	3	154374	9	2	6	0	4	1	
48839	58	3	151910	9	6	0	4	4	0	
48840	22	3	201490	9	4	0	3	4	1	
48841	52	4	287927	9	2	3	5	4	0	

46739 rows × 14 columns

In [41]:
```
x=data.drop(columns=['income'])
y=data['income']
```

In [42]:
```
x
```

Out[42]:

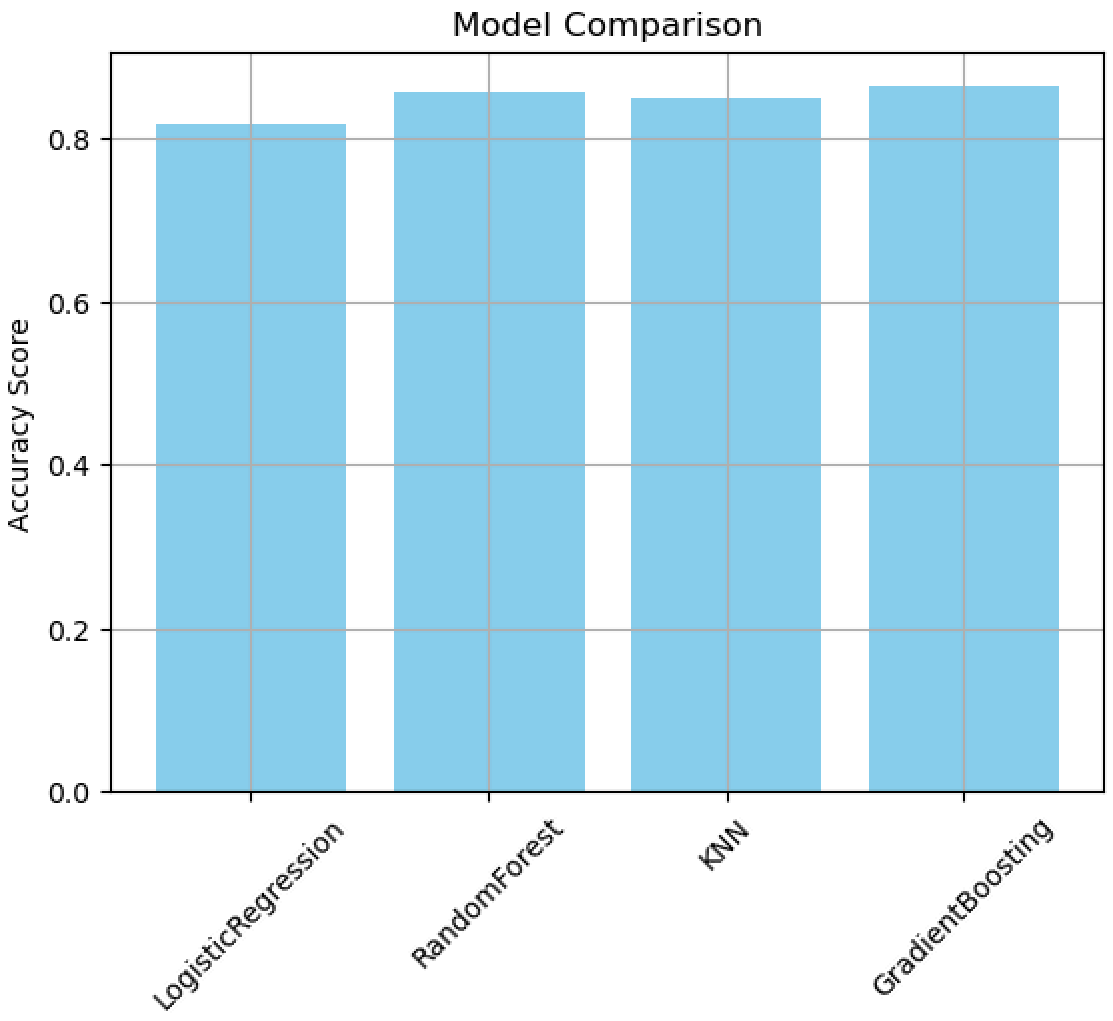
	age	workclass	fnlwgt	educational-num	marital-status	occupation	relationship	race	gender	ca
0	25	3	226802	7	4	6	3	2	1	
1	38	3	89814	9	2	4	0	4	1	
2	28	1	336951	12	2	11	0	4	1	
3	44	3	160323	10	2	6	0	2	1	
4	18	2	103497	10	4	8	3	4	0	
...	...	...	...	...	...	...	...	...	...	
48837	27	3	257302	12	2	13	5	4	0	
48838	40	3	154374	9	2	6	0	4	1	
48839	58	3	151910	9	6	0	4	4	0	
48840	22	3	201490	9	4	0	3	4	1	
48841	52	4	287927	9	2	3	5	4	0	

46739 rows × 13 columns

In [43]:
```
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, GradientBoostingClassifie
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, randc

models = {
    "LogisticRegression": LogisticRegression(),
    "RandomForest": RandomForestClassifier(),
    "KNN": 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}")
    print(classification_report(y_test, y_pred))
```

```
(name) Accuracy: 0.8177
              precision    recall  f1-score   support

       <=50K       0.84      0.94      0.89      7028
        >50K       0.71      0.45      0.55      2320

    accuracy                           0.82      9348
   macro avg       0.77      0.70      0.72      9348
weighted avg       0.81      0.82      0.80      9348

(name) Accuracy: 0.8580
              precision    recall  f1-score   support

       <=50K       0.89      0.93      0.91      7028
        >50K       0.75      0.64      0.69      2320

    accuracy                           0.86      9348
   macro avg       0.82      0.78      0.80      9348
weighted avg       0.85      0.86      0.85      9348

(name) Accuracy: 0.8493
              precision    recall  f1-score   support

       <=50K       0.87      0.95      0.90      7028
        >50K       0.77      0.56      0.65      2320

    accuracy                           0.85      9348
   macro avg       0.82      0.75      0.78      9348
weighted avg       0.84      0.85      0.84      9348

(name) Accuracy: 0.8639
              precision    recall  f1-score   support

       <=50K       0.88      0.95      0.91      7028
        >50K       0.80      0.60      0.69      2320

    accuracy                           0.86      9348
   macro avg       0.84      0.78      0.80      9348
weighted avg       0.86      0.86      0.86      9348
```

In [44]:
```python
import matplotlib.pyplot as plt

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



In [45]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifie
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

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

models = {
    "LogisticRegression": LogisticRegression(max_iter=1000),
    "RandomForest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC(),
    "GradeBoosting": GradientBoostingClassifier()
}

results = {}

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}")


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_nan


joblib.dump(best_model, "best_model.pkl")
print("Saved best model as best_model.pkl")
```

```
LogisticRegression: 0.7760
RandomForest: 0.8542
KNN: 0.7762
SVM: 0.7931
GradeBoosting: 0.8649

 Best model: GradeBoosting with accuracy 0.8649
Saved best model as best_model.pkl
```

In [46]:
```
%%writefile app.py
import streamlit as st
import pandas as pd
import joblib

model = joblib.load("best_model.pkl")

st.set_page_config(page_title="Employee Salary Classification", page_icon" ",

st.title(" Employee Salary Classification App")
st.markdown("Predict whether an employee earns >50K or ≤50K based on input fea

st.sidebar.header("Input Employee Details")

age = st.sidebar.slider("Age", 18, 65, 30)
education = st.sidebar.selectbox("Education Level", [
    "Bachelors", "Masters", "PhD", "HS-grad", "Assoc", "Some-college"
])
occupation = st.sidebar.selectbox("Job Role", [
    "Tech-support", "Craft-repair", "Other-service", "Sales",
    "Exec-managerial","Prof-speciality", "Handlers-cleaners", "Machine-op-insp
    "Adm-clerical" "Farming-fishing", "Transport-moving", "Priv-house-serv",
    "Protective-serv", "Armed-Forces"
])
hours_per_week = st.sidebar.slider("Hours per week", 1, 80, 40)
experience = st.sidebar.slider("Years of Experience", 0, 40, 5)

input_df = pd.DataFrame({
    'age': [age],
    'education': [education],
    'occupation': [occupation],
    'hours-per-week': [hours-per-week],
    'experience':[experience]
})

input_df = pd.DataFrame({
    'age': [age],
    'education': [education]
    'occupation': [occupation]
    'hours-per-week': [hours-per-week]
    'experience': [experience]
})

st.write("### Input Class")
st.write(input_df)

if st.button("Predict Salary Class"):
    prediction = model.predict(input_df)
    st.success(f" Prediction: {prediction[0]}")

st.markdown("....")
st.markdown("####  Batch Prediction")
uploaded_file = st.file_uploaded("Upload a CSV file for batch prediction", typ

if uploaded_file is not None:
    batch_data = pd.read_csv(uploaded_file)
    st.write("Uploaded data preview:", batch_data.head())
    batch_preds = model.predict(batch_data)
    batch_data['PredictedClass'] = batch_preds
    st.write("Predictions:")
    st.write(batch_data.head())
    csv = batch_data.to_csv(index=False).encode('utf-8')
    st.download_button("Download Preditions CSV", csv, file_name='predicted_cl
```

```
Overwriting app.py
```

In [47]:
```
y
```

Out[47]:
```
0        <=50K
1        <=50K
2         >50K
3         >50K
4        <=50K
         ...
48837    <=50K
48838     >50K
48839    <=50K
48840    <=50K
48841     >50K
Name: income, Length: 46739, dtype: object
```

In [48]:
```
!pip install streamlit pyngrok
```

```
Requirement already satisfied: streamlit in c:\users\lavan\anaconda3\lib\sit
e-packages (1.47.0)
Requirement already satisfied: pyngrok in c:\users\lavan\anaconda3\lib\site-
packages (7.2.12)
Requirement already satisfied: altair<6,>=4.0 in c:\users\lavan\anaconda3\li
b\site-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in c:\users\lavan\anaconda
3\lib\site-packages (from streamlit) (6.1.0)
Requirement already satisfied: click<9,>=7.0 in c:\users\lavan\anaconda3\lib
\site-packages (from streamlit) (8.0.4)
Requirement already satisfied: numpy<3,>=1.23 in c:\users\lavan\anaconda3\li
b\site-packages (from streamlit) (1.24.3)
Requirement already satisfied: packaging<26,>=20 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (23.1)
Requirement already satisfied: pandas<3,>=1.4.0 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (2.0.3)
Requirement already satisfied: pillow<12,>=7.1.0 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (9.4.0)
Requirement already satisfied: protobuf<7,>=3.20 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (6.31.1)
Requirement already satisfied: pyarrow>=7.0 in c:\users\lavan\anaconda3\lib
\site-packages (from streamlit) (11.0.0)
Requirement already satisfied: requests<3,>=2.27 in c:\users\lavan\anaconda3
\lib\site-packages (from streamlit) (2.31.0)
Requirement already satisfied: tenacity<10,>=8.1.0 in c:\users\lavan\anacond
a3\lib\site-packages (from streamlit) (8.2.2)
Requirement already satisfied: toml<2,>=0.10.1 in c:\users\lavan\anaconda3\l
ib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in c:\users\lavan
\anaconda3\lib\site-packages (from streamlit) (4.12.2)
Requirement already satisfied: watchdog<7,>=2.1.5 in c:\users\lavan\anaconda
3\lib\site-packages (from streamlit) (2.1.6)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\users\lava
n\anaconda3\lib\site-packages (from streamlit) (3.1.44)
Requirement already satisfied: pydeck<1,>=0.8.0b4 in c:\users\lavan\anaconda
3\lib\site-packages (from streamlit) (0.9.1)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in c:\users\lavan\a
naconda3\lib\site-packages (from streamlit) (6.3.2)
Requirement already satisfied: PyYAML>=5.1 in c:\users\lavan\anaconda3\lib\s
ite-packages (from pyngrok) (6.0)
Requirement already satisfied: jinja2 in c:\users\lavan\anaconda3\lib\site-p
ackages (from altair<6,>=4.0->streamlit) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in c:\users\lavan\anaconda3\l
ib\site-packages (from altair<6,>=4.0->streamlit) (4.17.3)
Requirement already satisfied: narwhals>=1.14.2 in c:\users\lavan\anaconda3
\lib\site-packages (from altair<6,>=4.0->streamlit) (1.47.1)
Requirement already satisfied: colorama in c:\users\lavan\anaconda3\lib\site
-packages (from click<9,>=7.0->streamlit) (0.4.6)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\lavan\anaconda3\l
ib\site-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.12)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\lavan\anac
onda3\lib\site-packages (from pandas<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\lavan\anaconda3\lib
\site-packages (from pandas<3,>=1.4.0->streamlit) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\lavan\anaconda3\li
b\site-packages (from pandas<3,>=1.4.0->streamlit) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\lavan\an
aconda3\lib\site-packages (from requests<3,>=2.27->streamlit) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\lavan\anaconda3\lib
```

```
\site-packages (from requests<3,>=2.27->streamlit) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\lavan\anaconda
3\lib\site-packages (from requests<3,>=2.27->streamlit) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\lavan\anaconda
3\lib\site-packages (from requests<3,>=2.27->streamlit) (2023.7.22)
Requirement already satisfied: smmap<6,>=3.0.1 in c:\users\lavan\anaconda3\l
ib\site-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->stream
lit) (5.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\lavan\anaconda3\l
ib\site-packages (from jinja2->altair<6,>=4.0->streamlit) (2.1.1)
Requirement already satisfied: attrs>=17.4.0 in c:\users\lavan\anaconda3\lib
\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (24.2.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0
in c:\users\lavan\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<
6,>=4.0->streamlit) (0.18.0)
Requirement already satisfied: six>=1.5 in c:\users\lavan\anaconda3\lib\site
-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.16.
0)
```

In [49]:
```
import os
import threading

def run_streamlit():
    os.system('streamlit run app.py --server.port 8501')

    thread = threading.Thread(target=run_streamlit)
    thread.start()
```

In [50]:
```
from pyngrok import ngrok
import time

time.sleep(5)

public_url = ngrok.connect(8501)
print("Your Streamlit app is live here:",public_url)
```

```
Installing ngrok ...
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pyngrok\installer.py:192, in install_ngro
k(ngrok_path, ngrok_version, **kwargs)
    190     download_path = _download_file(url, **kwargs)
--> 192     _install_ngrok_zip(ngrok_path, download_path)
    193 except Exception as e:

File ~\anaconda3\Lib\site-packages\pyngrok\installer.py:207, in _install_ngr
ok_zip(ngrok_path, zip_path)
    205 _print_progress("Installing ngrok ... ")
--> 207 with zipfile.ZipFile(zip_path, "r") as zip_ref:
    208     logger.debug(f"Extracting ngrok binary from {zip_path} to {ngrok
_path} ...")

File ~\anaconda3\Lib\zipfile.py:1284, in ZipFile.__init__(self, file, mode,
compression, allowZip64, compresslevel, strict_timestamps, metadata_encodin
g)
   1283 try:
-> 1284     self.fp = io.open(file, filemode)
   1285 except OSError:

OSError: [Errno 22] Invalid argument: 'C:\\Users\\lavan\\AppData\\Local\\Tem
p\\ngrok-v3-stable-windows-amd64.zip'

During handling of the above exception, another exception occurred:

PyngrokNgrokInstallError                  Traceback (most recent call last)
Cell In[50], line 6
      2 import time
      4 time.sleep(5)
----> 6 public_url = ngrok.connect(8501)
      7 print("Your Streamlit app is live here:",public_url)

File ~\anaconda3\Lib\site-packages\pyngrok\ngrok.py:385, in connect(addr, pr
oto, name, pyngrok_config, **options)
    381 _upgrade_legacy_params(pyngrok_config, options)
    383 logger.info(f"Opening tunnel named: {name}")
--> 385 api_url = get_ngrok_process(pyngrok_config).api_url
    387 logger.debug(f"Creating tunnel with options: {options}")
    389 tunnel = NgrokTunnel(api_request(f"{api_url}/api/tunnels", method="P
OST", data=options,
    390                                  timeout=pyngrok_config.request_time
out),
    391                      pyngrok_config, api_url)

File ~\anaconda3\Lib\site-packages\pyngrok\ngrok.py:201, in get_ngrok_proces
s(pyngrok_config)
    198 if pyngrok_config is None:
    199     pyngrok_config = conf.get_default()
--> 201 install_ngrok(pyngrok_config)
    203 return process.get_process(pyngrok_config)

File ~\anaconda3\Lib\site-packages\pyngrok\ngrok.py:129, in install_ngrok(py
ngrok_config)
    126     pyngrok_config = conf.get_default()
    128 if not os.path.exists(pyngrok_config.ngrok_path):
--> 129     installer.install_ngrok(pyngrok_config.ngrok_path, ngrok_version
=pyngrok_config.ngrok_version)
    131 config_path = conf.get_config_path(pyngrok_config)
    133 # Install the config to the requested path
```

```
File ~\anaconda3\Lib\site-packages\pyngrok\installer.py:194, in install_ngro
k(ngrok_path, ngrok_version, **kwargs)
    192     _install_ngrok_zip(ngrok_path, download_path)
    193 except Exception as e:
--> 194     raise PyngrokNgrokInstallError(f"An error occurred while downloa
ding ngrok from {url}: {e}")

PyngrokNgrokInstallError: An error occurred while downloading ngrok from htt
ps://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-windows-amd64.zip: (http
s://bin.equinox.io/c/bNyj1mQVY4c/ngrok-v3-stable-windows-amd64.zip:) [Errno
22] Invalid argument: 'C:\\Users\\lavan\\AppData\\Local\\Temp\\ngrok-v3-stab
le-windows-amd64.zip'
```

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
x=scaler.fit_transform(x)
x
```

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

In [ ]:
```
xtrain
```

In [ ]:
```
#machine learning algorithm
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(xtrain, ytrain)
predict=knn.predict(xtest)
predict
```

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

In [ ]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(xtrain, ytrain)
predict1=lr.predict(xtest)
predict1
```

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

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

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

In [ ]:

In [ ]:

In [ ]: