Lab Task:

10

Code Implementation:

Answer:

1) Libraries

```
2) import pandas as pd
3) import numpy as np
4) import pickle
5) from xgboost import XGBRegressor
6) from sklearn import metrics
7)
8) from sklearn.metrics import accuracy_score
9) from sklearn.preprocessing import LabelEncoder
10)from sklearn.model_selection import train_test_split
```

2) Data Loading And Reading:

```
df = pd.read_csv("data.csv")
df
```

3) Data Exploaration:

```
print(f"Number of Rows: {df.shape[0]} \nNumber of Columns: {df.shape[1]}")
```

```
df.head(2)
```

```
df.tail(2)
```

```
df.describe()
```

```
df.info()
```

```
print("-- Attributes in Data --")
for cols in df.columns:
    print(cols)
```

```
print("-- Number of instances in Data --")
```

Lab Task:

```
print(df.count())
```

```
df['city'].unique()
df.nunique()
```

```
print("-- Number of Null Values in Data --")
print(df.isnull().sum())
```

4) Data pre-processing:

```
df = df.drop('date', axis=1)
```

```
df.head(2)
```

```
df.info()
```

```
df['street'].mode()[0]
```

```
# df['street'] = df['street'].fillna(df['street'].mode()[0])
def fillNaObjMode(cols):
  for i in cols:
        df[i] = df[i].fillna(df[i].mode()[0])

columns = ['street','city','statezip','country']
fillNaObjMode(columns)
```

```
def fillNaIntMode(cols):
    for i in cols:
        df[i] = df[i].fillna(df[i].mode()[0])

columns = ['bedrooms','bathrooms','floors','waterfront','view','yr_built']
fillNaIntMode(columns)
```

```
def fillNaFloat(cols):
    for i in cols:
        df[i] = df[i].fillna(df[i].mean())

columns = ['price','sqft_living','sqft_lot','sqft_above','sqft_basement']
fillNaFloat(columns)
```

```
# df['price'] = df['price'].astype('int64')

def convertFloatintoInt(cols):
    for i in cols:
        df[i] = df[i].astype('int64')

columns =
['bedrooms','bathrooms','floors','waterfront','view','yr_built','price','sqft_living','sqft_lot','sqft_above','sqft_basement']
convertFloatintoInt(columns)
```

```
df.info()
```

```
df['street'].nunique()
```

```
df['country'].nunique()
```

```
df = df.drop('street',axis=1)
df = df.drop('country',axis=1)
```

```
def dataEncoder(cols):
    for i in cols:
        dataLabelEncoder = LabelEncoder()
        df[i] = dataLabelEncoder.fit_transform(df[i])

columns = ['city','statezip']
dataEncoder(columns)
```

```
df.to_csv(r'encoded-data.csv', index = False, header = True)
```

5) Train-Test-split:

```
X = df.drop(columns='bedrooms', axis=1)
Y = df['bedrooms']
```

```
print(X)
```

```
print(Y)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=2)
```

```
print(X.shape, X_train.shape, X_test.shape)
```

6) Model Apply & Classifier application;

```
regressor = XGBRegressor()
```

```
regressor.fit(X_train, Y_train)
```

```
training_data_prediction = regressor.predict(X_train)
```

```
r2_train = metrics.r2_score(Y_train, training_data_prediction)
```

```
print('R Squared value = ', r2_train)
```

```
test_data_prediction = regressor.predict(X_test)
```

```
r2_test = metrics.r2_score(Y_test, test_data_prediction)
```

```
print('R Squared value = ', r2_test)
```