```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

```
df=pd.read_csv('/content/data (1).csv')
df
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_baseme
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	2
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	10
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	8
...	...	...	...	...	...	...	...	...	...	...	...	
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	10
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	

4600 rows × 18 columns

```
df.dtypes
```

```
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
lst =['date','street','city','statezip','country']

for i in lst:

    if df[i].apply(type).nunique() > 1:

        print(f"Column '{i}' contains mixed types. Handling required.")

        df[i] = df[i].astype(str)

    df[i] = le.fit_transform(df[i])
```

```
df.dtypes
```

```
date               int64
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street             int64
city               int64
statezip           int64
country            int64
dtype: object
```

```
X = df.drop(columns=['price'])
y = df['price']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor(),
    'Decision Tree': DecisionTreeRegressor()
}

for name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)

    mse = mean_squared_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)

    print(f"{name}:")
    print(f"  Mean Squared Error: {mse}")
    print(f"  R^2 Score: {r2}")
    print()
```

```
Linear Regression:
  Mean Squared Error: 986145473005.4001
  R^2 Score: 0.0330450439020854

Random Forest:
  Mean Squared Error: 977117444900.4146
  R^2 Score: 0.04189738542660937

Gradient Boosting:
  Mean Squared Error: 967564600405.6816
  R^2 Score: 0.05126433034687883

Decision Tree:
  Mean Squared Error: 1019526247692.7129
  R^2 Score: 0.0003138633553505521
```