Nifty 50 price prediction using past 25 years data from 1999 to 2024

In [1]:

```
#importing required libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
#importing dataset
dataset=pd.read_csv("nifty_50_15_year_data.csv")
dataset.head(3)
```

Out[2]:

	Index Name	Date	Open	High	Low	Close
0	NIFTY 50	23 Oct 2024	24378.15	24604.25	24378.1	24435.5
1	NIFTY 50	22 Oct 2024	24798.65	24882.00	24445.8	24472.1
2	NIFTY 50	21 Oct 2024	24956.15	24978.30	24679.6	24781.1

In [3]:

```
dataset.shape
```

Out[3]:

```
(6219, 6)
```

In [4]:

```
dataset_1=pd.DataFrame(dataset.iloc[:,-1:])
dataset_1.head(3)
```

Out[4]:

	Close
0	24435.5
1	24472.1
2	24781.1

In [5]:

```
#we will predict by using previous day's high, low, close and current day's open
```

In [6]:

```
prev_high=dataset["High"].shift(periods=-1)
prev_low=dataset["Low"].shift(periods=-1)
prev_close=dataset["Close"].shift(periods=-1)
```

In [7]:

```
dataset=dataset.drop(["Index Name","Date","Low","High","Close"], axis=1)
dataset.head(6)
```

Out[7]:

	Open
0	24378.15
1	24798.65
2	24956.15
3	24664.95
4	25027.40
5	25008.55

In [8]:

```
dataset= pd.concat([dataset, prev_high,prev_low,prev_close], axis=1,join="inner")
dataset.head(3)
```

Out[8]:

	Open	High	Low	Close
0	24378.15	24882.0	24445.80	24472.10
1	24798.65	24978.3	24679.60	24781.10
2	24956.15	24886.2	24567.65	24854.05

In [9]:

```
dataset.rename(columns={"High": "Prev_High",
                        "Low":"Prev_Low",
                        "Close":"Prev_Close"},inplace=True)
dataset.head(3)
```

Out[9]:

	Open	Prev_High	Prev_Low	Prev_Close
0	24378.15	24882.0	24445.80	24472.10
1	24798.65	24978.3	24679.60	24781.10
2	24956.15	24886.2	24567.65	24854.05

In [10]:

```
dataset= pd.concat([dataset, dataset_1], axis=1,join="inner")
dataset.head(3)
```

Out[10]:

	Open	Prev_High	Prev_Low	Prev_Close	Close
0	24378.15	24882.0	24445.80	24472.10	24435.5
1	24798.65	24978.3	24679.60	24781.10	24472.1
2	24956.15	24886.2	24567.65	24854.05	24781.1

In [11]:

```
#finding null values
dataset.isnull().sum()
```

Out[11]:

```
Open          0
Prev_High     1
Prev_Low      1
Prev_Close    1
Close         0
dtype: int64
```

In [12]:

```python
#removing null values using backward filling
dataset["Prev_High"].fillna(dataset["Prev_High"].mode()[0],axis=0,inplace=True)
dataset["Prev_Low"].fillna(dataset["Prev_Low"].mode()[0],axis=0,inplace=True)
dataset["Prev_Close"].fillna(dataset["Prev_Close"].mode()[0],axis=0,inplace=True)
dataset.isnull().sum()
```

```
C:\Users\Aryansh Pathak\AppData\Local\Temp\ipykernel_6844\4175173473.py:2: FutureWar
ning: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["Prev_High"].fillna(dataset["Prev_High"].mode()[0],axis=0,inplace=True)
C:\Users\Aryansh Pathak\AppData\Local\Temp\ipykernel_6844\4175173473.py:3: FutureWar
ning: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["Prev_Low"].fillna(dataset["Prev_Low"].mode()[0],axis=0,inplace=True)
C:\Users\Aryansh Pathak\AppData\Local\Temp\ipykernel_6844\4175173473.py:4: FutureWar
ning: A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  dataset["Prev_Close"].fillna(dataset["Prev_Close"].mode()[0],axis=0,inplace=True)
```

Out[12]:

```
Open          0
Prev_High     0
Prev_Low      0
Prev_Close    0
Close         0
dtype: int64
```

In [13]:

```
#finding if any duplicate value is present in the dataset
dataset.duplicated().sum()
```

Out[13]:

```
np.int64(0)
```

In [14]:

```
dataset.head(3)
```

Out[14]:

	Open	Prev_High	Prev_Low	Prev_Close	Close
0	24378.15	24882.0	24445.80	24472.10	24435.5
1	24798.65	24978.3	24679.60	24781.10	24472.1
2	24956.15	24886.2	24567.65	24854.05	24781.1

In [15]:

```
#scaling data using Standard_scaler
x=dataset[["Open","Prev_High","Prev_Low","Prev_Close","Close"]]
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
ss.fit_transform(x)
dataset=pd.DataFrame(ss.fit_transform(x))
dataset.head(3)
```

Out[15]:

	0	1	2	3	4
0	2.914794	2.986537	2.952708	2.936027	2.927356
1	2.986971	3.003023	2.993065	2.989134	2.933643
2	3.014005	2.987256	2.973741	3.001672	2.986719

In [16]:

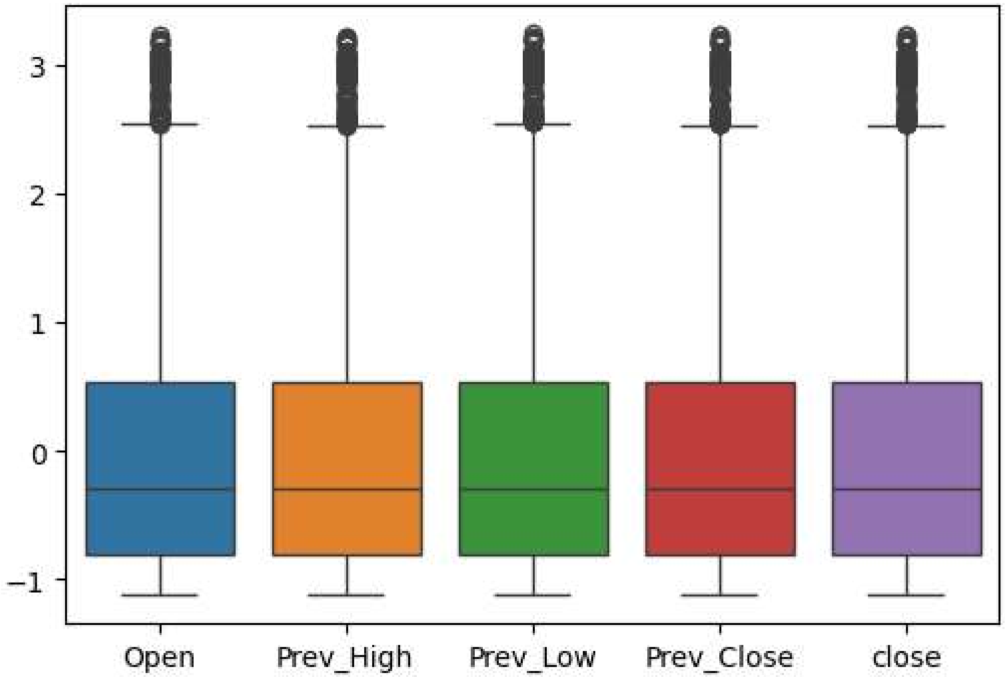
```
dataset=dataset.rename(columns={0:  "Open",
                                1:  "Prev_High",
                                2:  "Prev_Low",
                                3:  "Prev_Close",
                                4:  "close"
                                }
                      )
dataset.head(3)
```

Out[16]:

	Open	Prev_High	Prev_Low	Prev_Close	close
0	2.914794	2.986537	2.952708	2.936027	2.927356
1	2.986971	3.003023	2.993065	2.989134	2.933643
2	3.014005	2.987256	2.973741	3.001672	2.986719

In [17]:

```
plt.figure(figsize=(6,4))
sns.boxplot(data=dataset)
plt.show()
```



In [18]:

```
dataset.isnull().sum()
```

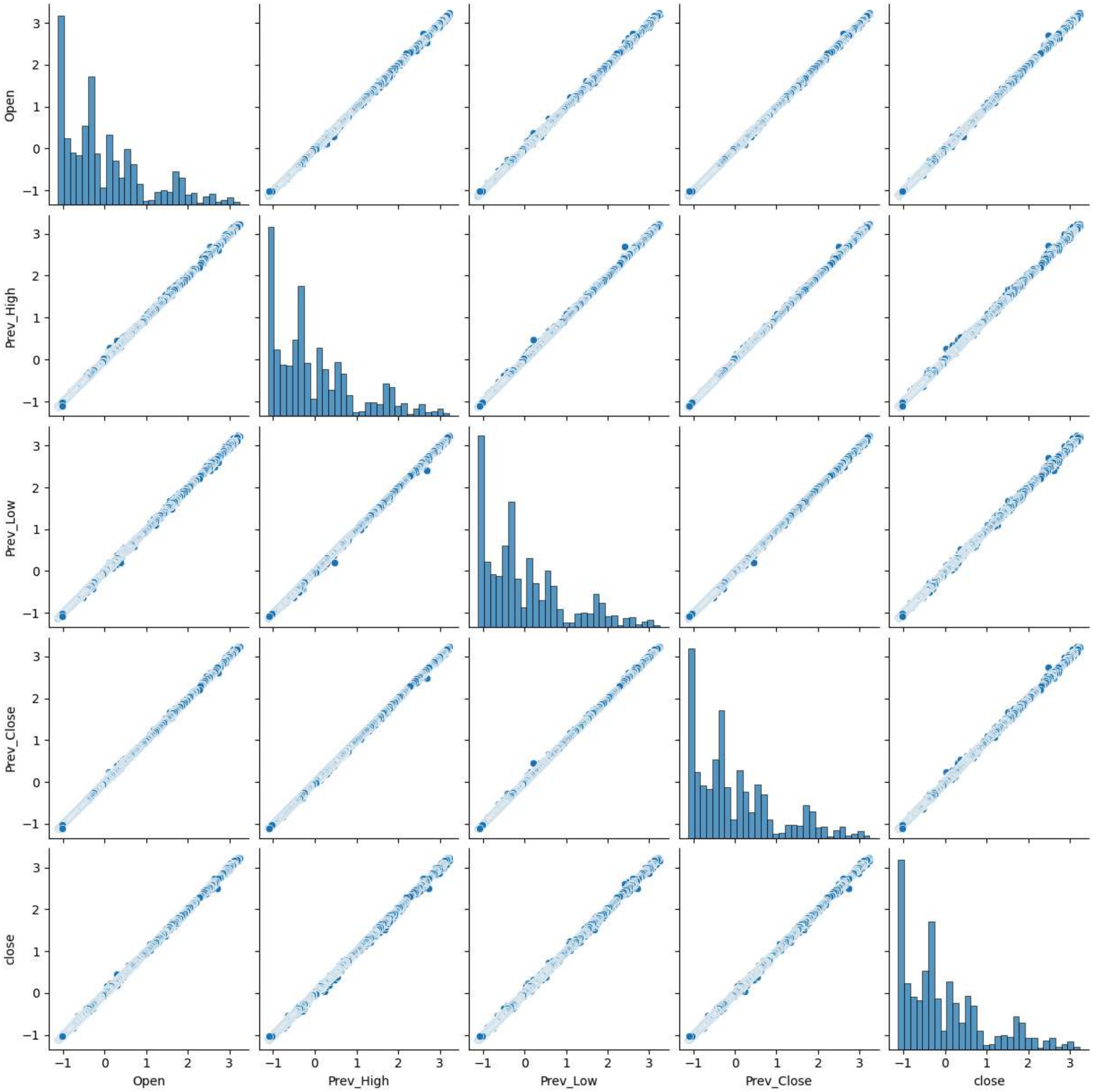
Out[18]:

```
Open          0
Prev_High     0
Prev_Low      0
Prev_Close    0
close         0
dtype: int64
```

In [19]:

```
plt.figure(figsize=(4,4))
sns.pairplot(dataset)
plt.show()
```

```
<Figure size 400x400 with 0 Axes>
```



In [20]:
```
#by seeing graph we can infer that linear regression will work very accurately
#for finding the best accuracy
```

In [21]:
```
x=dataset.iloc[:,:-1]
y=dataset.iloc[:,-1:]
```

In [22]:
```
#applying train_test_split
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

In [23]:
```
#applying linear regression model
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
lr.score(x_test,y_test)*100,lr.score(x_train,y_train)*100
```

Out[23]:
```
(99.98134713265982, 99.9807379794134)
```

In [24]:
```
lr.predict([[24956.15,24886.20,24567.65,24854.05]])
```

```
C:\Users\Aryansh Pathak\AppData\Local\Programs\Python\Python312\Lib\site-packages\sk
learn\base.py:493: UserWarning: X does not have valid feature names, but LinearRegre
ssion was fitted with feature names
  warnings.warn(
```

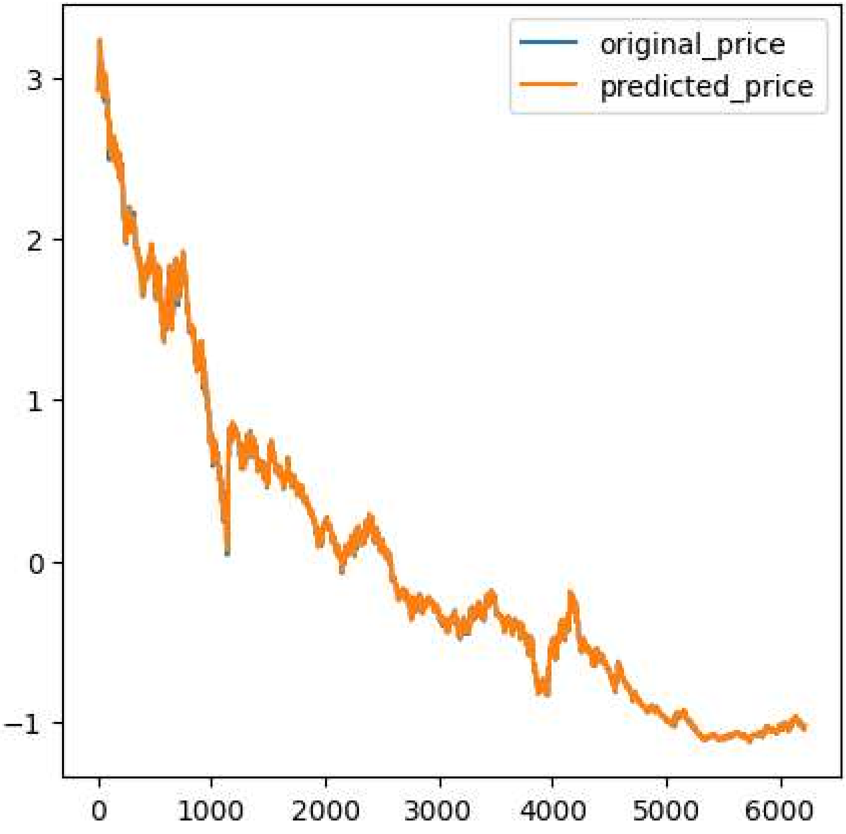
Out[24]:
```
array([[24971.10488591]])
```

In [25]:
```
#Conclusion- It can be inferred that some over fitting of data has occured as the a
             #hence we need to futhur optimise it.
```

In [26]:
```
#PLOTTING PREDICTED VALUE VS ORIGINAL VALUE
```

In [30]:
```
predicted=lr.predict(x)
original=y
```

In [31]:
```
plt.figure(figsize=(5,5))
plt.plot(original)
plt.plot(predicted)
plt.legend(["original_price","predicted_price"])
plt.show()
```



In [ ]: