# DowJones_

By Hatem Elgenedy

November 7, 2025

[ ]:
```
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
```

[82]:
```
print(sns.get_dataset_names())
```

```
['anagrams', 'anscombe', 'attention', 'brain_networks', 'car_crashes',
'diamonds', 'dots', 'dowjones', 'exercise', 'flights', 'fmri', 'geyser', 'glue',
'healthexp', 'iris', 'mpg', 'penguins', 'planets', 'seaice', 'taxis', 'tips',
'titanic']
```

[83]:
```
df = sns.load_dataset("dowjones")
df.head()
```

[83]:

| | Date | Price |
|---|---|---|
| 0 | 1914-12-01 | 55.00 |
| 1 | 1915-01-01 | 56.55 |
| 2 | 1915-02-01 | 56.00 |
| 3 | 1915-03-01 | 58.30 |
| 4 | 1915-04-01 | 66.45 |

[84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 649 entries, 0 to 648
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    649 non-null    datetime64[ns]
 1   Price   649 non-null    float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 10.3 KB
```

[85]:
```
df.shape
```

[85]:
```
(649, 2)
```

[86]:
```
df.isnull().sum()
```

[86]:
```
Date     0
Price    0
dtype: int64
```

[87]:
```
df.Date
```

[87]:
```
0     1914-12-01
1     1915-01-01
2     1915-02-01
3     1915-03-01
4     1915-04-01
          …
644   1968-08-01
645   1968-09-01
646   1968-10-01
647   1968-11-01
648   1968-12-01
Name: Date, Length: 649, dtype: datetime64[ns]
```

[88]:
```
df.duplicated().sum()
```

[88]:
```
0
```

[89]:
```
df.Price
```

[89]:
```
0       55.00
1       56.55
2       56.00
3       58.30
4       66.45
         …
644    883.72
645    922.80
646    955.47
647    964.12
648    965.39
Name: Price, Length: 649, dtype: float64
```

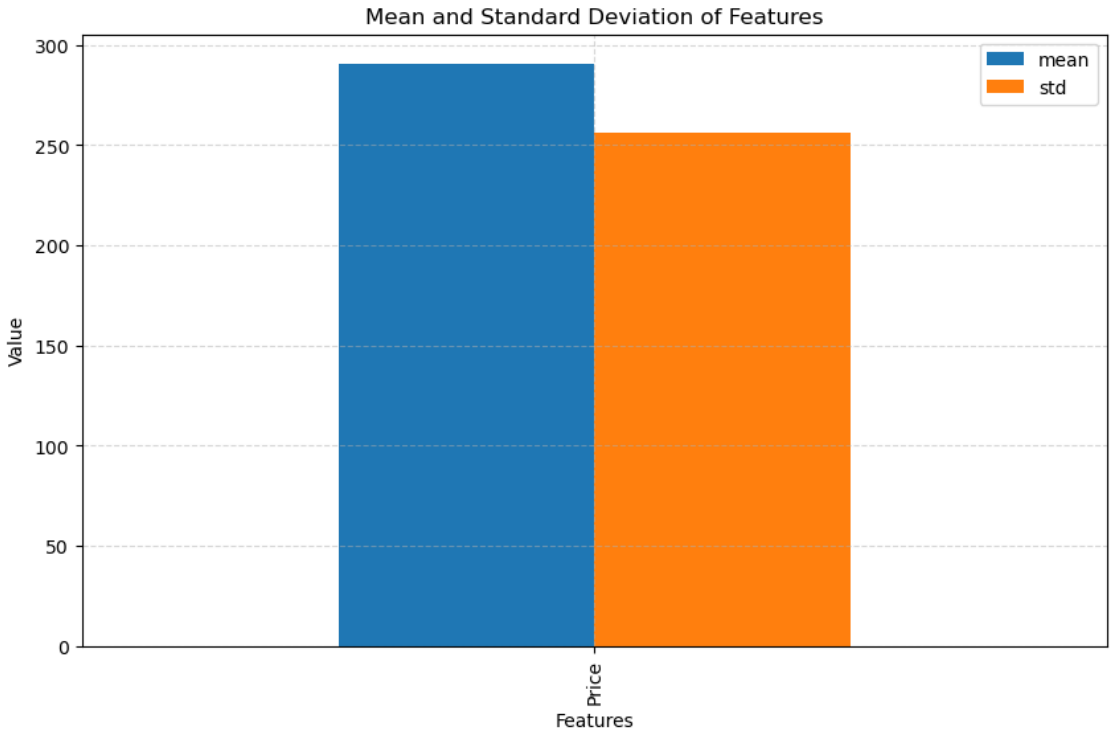
[90]:
```
df.describe()
```

[90]:

| | Date | Price |
|---|---|---|
| count | 649 | 649.000000 |
| mean | 1941-11-30 17:42:48.258859776 | 290.807319 |
| min | 1914-12-01 00:00:00 | 46.850000 |
| 25% | 1928-06-01 00:00:00 | 106.900000 |
| 50% | 1941-12-01 00:00:00 | 172.270000 |
| 75% | 1955-06-01 00:00:00 | 436.730000 |
| max | 1968-12-01 00:00:00 | 985.930000 |

```
std                                NaN  256.062906
```

[195]:
```
desc = df.describe().T

desc[['mean', 'std']].plot(kind='bar', figsize=(10,6))
plt.title('Mean and Standard Deviation of Features')
plt.xlabel('Features')
plt.ylabel('Value')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```




[91]:
```
len(df.Price)
```

[91]:
```
649
```

[92]:
```
len(df.Date)
```

[92]:
```
649
```

[93]:
```
df.Date.value_counts()
```

[93]:
```
Date
1914-12-01    1
```

```
1952-02-01    1
1950-08-01    1
1950-09-01    1
1950-10-01    1
              ..
1933-02-01    1
1933-03-01    1
1933-04-01    1
1933-05-01    1
1968-12-01    1
Name: count, Length: 649, dtype: int64
```

[94]:
```
df.Price.value_counts()
```

[94]:
```
Price
98.50     2
97.00     2
120.10    2
99.45     2
75.70     2
          ..
102.85    1
106.90    1
102.30    1
96.20     1
965.39    1
Name: count, Length: 637, dtype: int64
```

[95]:
```
!pip install ydata-profiling
```

```
Requirement already satisfied: ydata-profiling in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (4.17.0)
Requirement already satisfied: scipy<1.16,>=1.4.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (1.13.1)
Requirement already satisfied: pandas!=1.4.0,<3.0,>1.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (2.3.1)
Requirement already satisfied: matplotlib<=3.10,>=3.5 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (3.10.0)
Requirement already satisfied: pydantic>=2 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (2.11.7)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (6.0.2)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
```

```
profiling) (3.1.6)
Requirement already satisfied: visions<0.8.2,>=0.7.5 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
visions[type_image_path]<0.8.2,>=0.7.5->ydata-profiling) (0.8.1)
Requirement already satisfied: numpy<2.2,>=1.16.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (1.26.4)
Requirement already satisfied: minify-html>=0.15.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (0.18.1)
Requirement already satisfied: filetype>=1.0.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (1.2.0)
Requirement already satisfied: phik<0.13,>=0.11.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (0.12.5)
Requirement already satisfied: requests<3,>=2.24.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (2.32.4)
Requirement already satisfied: tqdm<5,>=4.48.2 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (4.67.1)
Requirement already satisfied: seaborn<0.14,>=0.10.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (0.13.2)
Requirement already satisfied: multimethod<2,>=1.4 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (1.12)
Requirement already satisfied: statsmodels<1,>=0.13.2 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (0.14.5)
Requirement already satisfied: typeguard<5,>=3 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (4.4.4)
Requirement already satisfied: imagehash==4.3.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (4.3.1)
Requirement already satisfied: wordcloud>=1.9.3 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (1.9.4)
Requirement already satisfied: dacite>=1.8 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from ydata-profiling) (1.9.2)
Requirement already satisfied: numba<=0.61,>=0.56.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from ydata-
profiling) (0.61.0)
Requirement already satisfied: PyWavelets in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from imagehash==4.3.1->ydata-profiling)
(1.9.0)
Requirement already satisfied: pillow in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from imagehash==4.3.1->ydata-profiling)
(11.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from jinja2<3.2,>=2.11.1->ydata-profiling)
```

```
(3.0.2)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from matplotlib<=3.10,>=3.5->ydata-profiling)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from matplotlib<=3.10,>=3.5->ydata-profiling)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
matplotlib<=3.10,>=3.5->ydata-profiling) (4.55.3)
Requirement already satisfied: kiwisolver>=1.3.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
matplotlib<=3.10,>=3.5->ydata-profiling) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from matplotlib<=3.10,>=3.5->ydata-profiling)
(24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from matplotlib<=3.10,>=3.5->ydata-profiling)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
matplotlib<=3.10,>=3.5->ydata-profiling) (2.9.0.post0)
Requirement already satisfied: llvmlite<0.45,>=0.44.0dev0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
numba<=0.61,>=0.56.0->ydata-profiling) (0.44.0)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from pandas!=1.4.0,<3.0,>1.1->ydata-profiling)
(2025.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from pandas!=1.4.0,<3.0,>1.1->ydata-profiling)
(2025.2)
Requirement already satisfied: joblib>=0.14.1 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from phik<0.13,>=0.11.1->ydata-profiling)
(1.5.1)
Requirement already satisfied: charset_normalizer<4,>=2 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.24.0->ydata-profiling) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from requests<3,>=2.24.0->ydata-profiling)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.24.0->ydata-profiling) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
requests<3,>=2.24.0->ydata-profiling) (2025.8.3)
Requirement already satisfied: patsy>=0.5.6 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from statsmodels<1,>=0.13.2->ydata-profiling)
```

```
(1.0.2)
Requirement already satisfied: typing_extensions>=4.14.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
typeguard<5,>=3->ydata-profiling) (4.15.0)
Requirement already satisfied: attrs>=19.3.0 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from
visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5->ydata-profiling)
(24.3.0)
Requirement already satisfied: networkx>=2.4 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from
visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5->ydata-profiling)
(3.4.2)
Requirement already satisfied: puremagic in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from
visions<0.8.2,>=0.7.5->visions[type_image_path]<0.8.2,>=0.7.5->ydata-profiling)
(1.30)
Requirement already satisfied: annotated-types>=0.6.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
pydantic>=2->ydata-profiling) (0.6.0)
Requirement already satisfied: pydantic-core==2.33.2 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
pydantic>=2->ydata-profiling) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in
/opt/anaconda3/envs/anaconda-nlp/lib/python3.11/site-packages (from
pydantic>=2->ydata-profiling) (0.4.0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/envs/anaconda-
nlp/lib/python3.11/site-packages (from python-
dateutil>=2.7->matplotlib<=3.10,>=3.5->ydata-profiling) (1.17.0)
```

[96]:
```
from ydata_profiling import ProfileReport
```

[97]:
```
profile = ProfileReport(df , title = "DowJones Profiling Report" , explorative␣
↪= True)
```

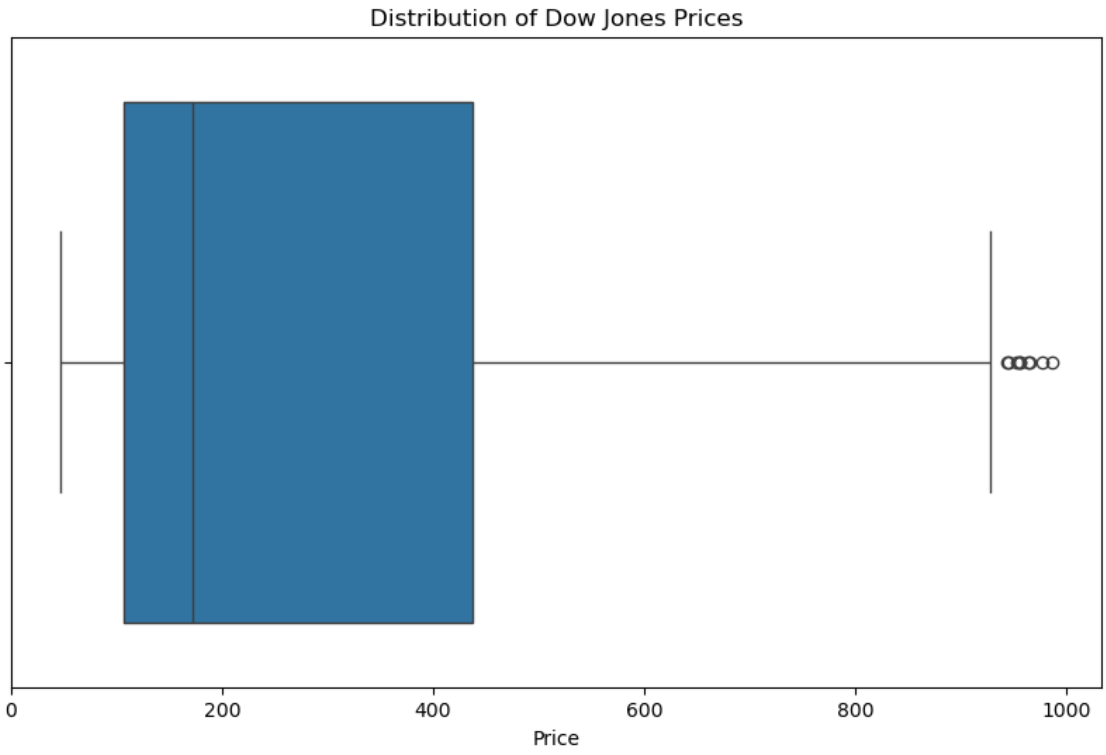
[98]:
```
profile.to_notebook_iframe()
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

100%|      | 2/2 [00:00<00:00, 1639.36it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[99]:
```
profile.to_file("DowJones_report.html")
```

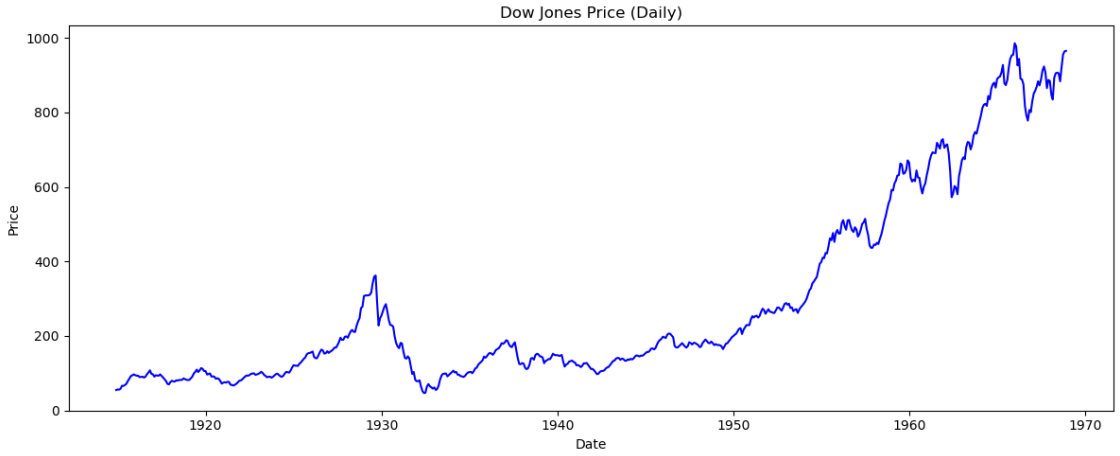
```
Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

[100]:
```
%matplotlib inline

fig = plt.figure(figsize=(10,6))
ax = fig.subplots()

sns.boxplot(x=df["Price"], ax=ax)
plt.title("Distribution of Dow Jones Prices")
plt.xlabel("Price")
plt.show()
```




[101]:
```
import matplotlib.dates as mdates
fig, ax = plt.subplots(figsize=(12,5))
ax.plot(df['Date'], df['Price'], color='blue')
ax.set_title('Dow Jones Price (Daily)')
ax.set_xlabel('Date')
ax.set_ylabel('Price')

locator = mdates.AutoDateLocator()
formatter = mdates.ConciseDateFormatter(locator)
ax.xaxis.set_major_locator(locator)
ax.xaxis.set_major_formatter(formatter)
```

```
plt.tight_layout()
plt.show()
```




[102]:
```
df1 = df.copy()
```

[103]:
```
df1['Price'].describe()
```

[103]:
```
count    649.000000
mean     290.807319
std      256.062906
min       46.850000
25%      106.900000
50%      172.270000
75%      436.730000
max      985.930000
Name: Price, dtype: float64
```

[104]:
```
q1 = df1['Price'].quantile(0.25)
q3 = df1['Price'].quantile(0.75)
print(q1,q3)
```

```
106.9 436.73
```

[105]:
```
IQR = q3 - q1
print(IQR)
```

```
329.83000000000004
```

[106]:
```
upper_limit = q3 +1.5 *IQR
lower_limit = q1 - 1.5 * IQR
```

[107]:
```
upper_limit
```

[107]:
```
931.4750000000001
```

[108]:
```
lower_limit
```

[108]:
```
-387.845
```

[109]:
```
df1[df1['Price'] > upper_limit]
```

[109]:
```
           Date   Price
610 1965-10-01  944.77
611 1965-11-01  953.31
612 1965-12-01  955.19
613 1966-01-01  985.93
614 1966-02-01  977.15
616 1966-04-01  943.70
646 1968-10-01  955.47
647 1968-11-01  964.12
648 1968-12-01  965.39
```

[110]:
```
len(df1[df1['Price']  > upper_limit])
```

[110]:
```
9
```

[111]:
```
df1.shape
```

[111]:
```
(649, 2)
```

[112]:
```
df2 =df1[(df['Price'] > lower_limit) & (df1['Price'] < upper_limit)]
```

[113]:
```
df2.shape
```

[113]:
```
(640, 2)
```

[114]:
```
df2.Price.value_counts()
```

[114]:
```
Price
98.50     2
97.00     2
99.45     2
75.70     2
90.65     2
         ..
91.65     1
95.45     1
99.05     1
102.85    1
922.80    1
Name: count, Length: 628, dtype: int64
```

[115]:
```
df2.columns
```

[115]:
```
Index(['Date', 'Price'], dtype='object')
```

[116]:
```
X =df2.drop(columns = ['Price'])
```

[117]:
```
X.columns
```

[117]:
```
Index(['Date'], dtype='object')
```

[118]:
```
X =pd.get_dummies(X , drop_first = True).astype('int')
```

[119]:
```
X.head()
```

[119]:

| | Date |
|---|---|
| 0 | -1738368000000000000 |
| 1 | -1735689600000000000 |
| 2 | -1733011200000000000 |
| 3 | -1730592000000000000 |
| 4 | -1727913600000000000 |

[120]:
```
X.shape
```

[120]:
```
(640, 1)
```

[121]:
```
y = df2[['Price']]
```

[122]:
```
y.columns
```

[122]:
```
Index(['Price'], dtype='object')
```

[123]:
```
y.shape
```

[123]:
```
(640, 1)
```

[124]:
```
from sklearn.model_selection import train_test_split as split
```

[125]:
```
X_train , X_test , y_train , y_test = split(X,y , train_size = 0.8 ,␣
↪random_state = 42)
```

[126]:
```
X_train.shape
```

[126]:
```
(512, 1)
```

[127]:
```
X_test.shape
```

[127]:
```
(128, 1)
```

[128]:
```
y_test.shape
```

[128]: (128, 1)

[129]:
```
y_train.shape
```

[129]: (512, 1)

[130]:
```
X.head()
```

[130]:
```
                   Date
0 -1738368000000000000
1 -1735689600000000000
2 -1733011200000000000
3 -1730592000000000000
4 -1727913600000000000
```

[131]:
```
y.head()
```

[131]:
```
   Price
0  55.00
1  56.55
2  56.00
3  58.30
4  66.45
```

[132]:
```
X.columns
```

[132]: Index(['Date'], dtype='object')

[133]:
```
y.columns
```

[133]: Index(['Price'], dtype='object')

[134]:
```
X_test.head()
```

[134]:
```
                     Date
570  -239414400000000000
265 -1041379200000000000
291  -973209600000000000
597  -168307200000000000
174 -1280793600000000000
```

[135]:
```
y_test.head()
```

[135]:
```
      Price
570  572.64
265  182.30
291  142.05
597  863.55
```

```
174  316.45
```

[136]:
```
from sklearn.preprocessing import StandardScaler
```

[137]:
```
sc = StandardScaler()
```

[138]:
```
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

[139]:
```
X_train[:10]
```

[139]:
```
array([[ 0.46501553],
       [-0.40327054],
       [ 0.57406695],
       [ 0.59038879],
       [ 0.79808871],
       [-0.77974906],
       [ 0.04997443],
       [ 0.86857754],
       [ 0.87951855],
       [ 1.50225957]])
```

[140]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[141]:
```
df2 = df2.sort_values('Date').reset_index(drop=True)
```

[142]:
```
df2['Pct_Change'] = df2['Price'].pct_change() * 100
```

[143]:
```
df2['Next_Pct'] = df2['Pct_Change'].shift(-1)
df2['Target'] = (df2['Next_Pct'] > 0).astype(int)
```

[144]:
```
df2 = df2.dropna()
```

[145]:
```
X = df2[['Pct_Change']]
y = df2['Target']
```

[146]:
```
X_train, X_test , y_train ,y_test = train_test_split(X ,  y , test_size = 0.2 ,␣
↪random_state = 42)
```

[147]:
```
from sklearn.linear_model import LogisticRegression
```

[148]:
```
model = LogisticRegression()
```

[149]:
```
model.fit(X_train , y_train)
```

[149]:
```
LogisticRegression()
```

[150]:
```
y_pred = model.predict(X_test)
```

[151]:
```
df2 = df2.copy()
```

[152]:
```
df2['MA_3'] = df2['Price'].rolling(3).mean()
df2['MA_7'] = df2['Price'].rolling(7).mean()
df2['Vol_5'] = df2['Pct_Change'].rolling(5).std()

df2 = df2.dropna()

X = df2[['Pct_Change', 'MA_3', 'MA_7', 'Vol_5']]
y = df2['Target']
```

[153]:
```
df2['Return_1'] = df2['Price'].pct_change(1)
df2['Return_5'] = df2['Price'].pct_change(5)

# Trend indicators
df2['MA_10'] = df2['Price'].rolling(10).mean()
df2['MA_20'] = df2['Price'].rolling(20).mean()
df2['MA_Ratio'] = df2['MA_10'] / df2['MA_20']


df2['Vol_10'] = df2['Return_1'].rolling(10).std()

# Lag feature
df2['Lag_1'] = df2['Return_1'].shift(1)
```

[154]:
```
features = ['Return_1', 'Return_5', 'MA_10', 'MA_20', 'MA_Ratio', 'Vol_10',␣
 ↪'Lag_1']
X = df2[features].dropna()
y = df2['Target'].loc[X.index]
```

[155]:
```
accuracy = accuracy_score(y_test ,  y_pred)
print("Model Accuracy:" , round(accuracy * 100,2) , "%")
```

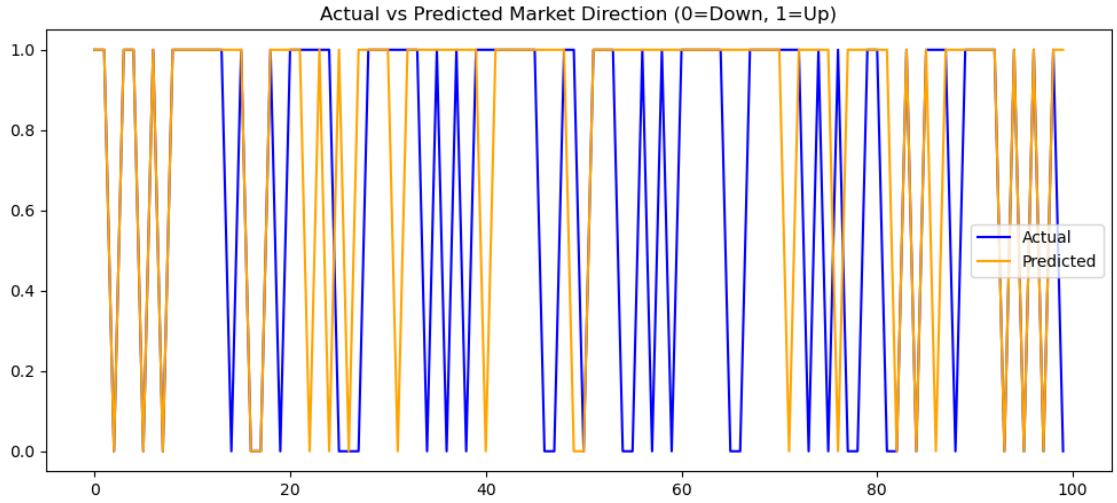
```
Model Accuracy: 68.75 %
```

[156]:
```
df2['MA_3'] = df2['Price'].rolling(3).mean()
df2['MA_7'] = df2['Price'].rolling(7).mean()
df2['Vo1_5'] = df2['Pct_Change'].rolling(5).std()
```

[157]:
```
X = df2[['Pct_Change' , 'MA_3' , 'MA_7' ,'Vo1_5']].dropna()
y = df2['Target'].loc[X.index]
```

[158]:
```
plt.figure(figsize=(12,5))
plt.plot(y_test.values[:100], label='Actual', color='blue')
plt.plot(y_pred[:100], label='Predicted', color='orange')
plt.title('Actual vs Predicted Market Direction (0=Down, 1=Up)')
```

```
plt.legend()
plt.show()
```




[162]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

[163]:
```
dfc = df2[['Date', 'Price']].copy()
dfc['Date'] = pd.to_datetime(dfc['Date'], errors='coerce')
dfc = dfc.dropna(subset=['Date', 'Price']).sort_values('Date').
 ↪reset_index(drop=True)


dfc['ret_1']   = dfc['Price'].pct_change(1)
dfc['ma_5']    = dfc['Price'].rolling(5).mean()
dfc['ma_10']   = dfc['Price'].rolling(10).mean()
dfc['vol_5']   = dfc['ret_1'].rolling(5).std()
dfc['lag_price_1'] = dfc['Price'].shift(1)
dfc['target_next_price'] = dfc['Price'].shift(-1)


dfc = dfc.
 ↪dropna(subset=['ret_1','ma_5','ma_10','vol_5','lag_price_1','target_next_price']).
 ↪reset_index(drop=True)
```

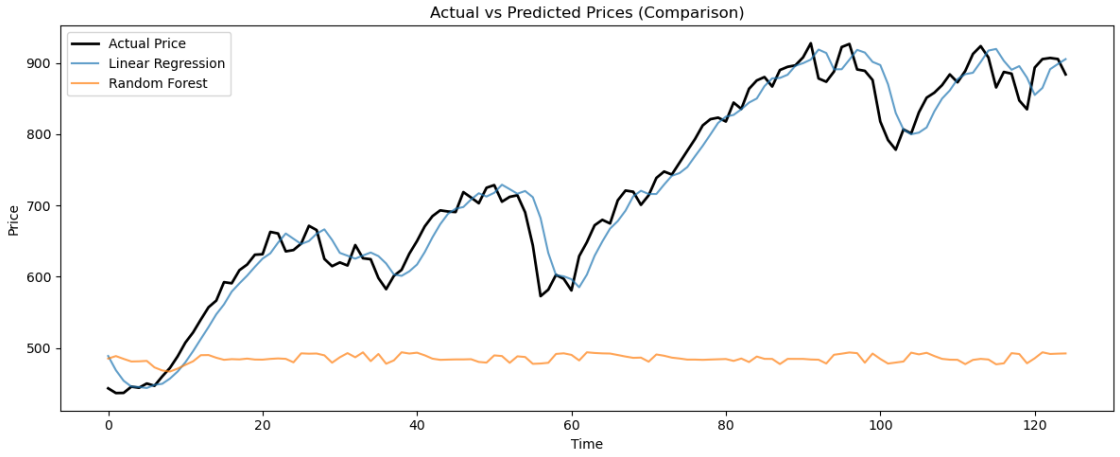
[164]:
```
X = dfc[['lag_price_1', 'ret_1', 'ma_5', 'ma_10', 'vol_5']]
y = dfc['target_next_price']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪shuffle=False)




linreg = make_pipeline(StandardScaler(), Ridge(alpha=1.0))
linreg.fit(X_train, y_train)
y_pred_lr = linreg.predict(X_test)


rf = RandomForestRegressor(n_estimators=300, max_depth=8, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

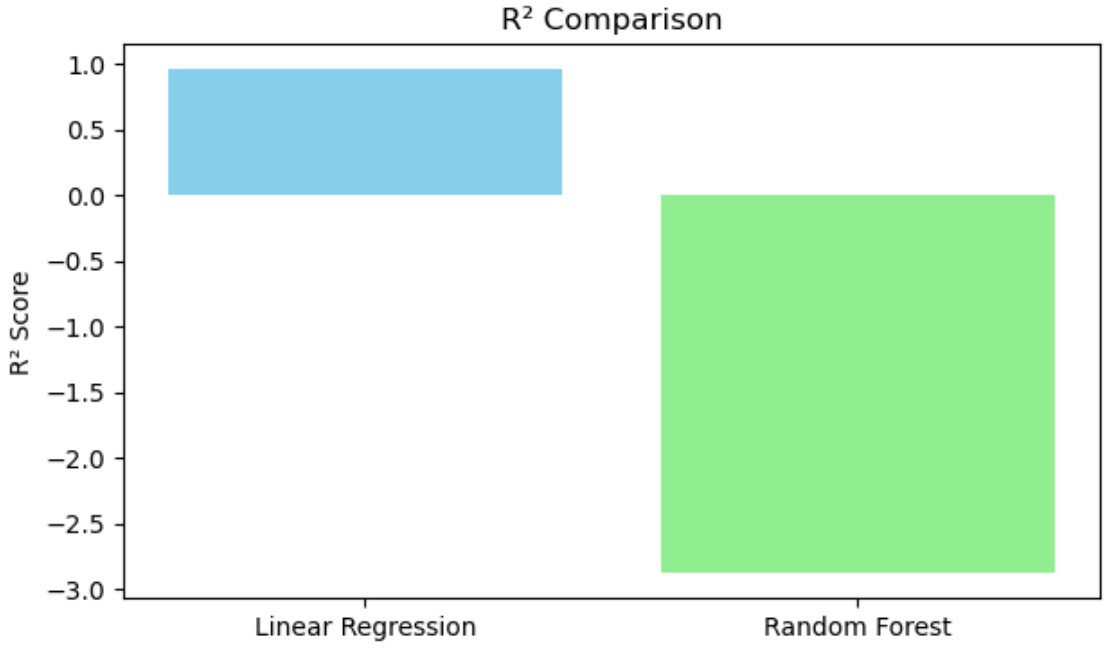
[165]:
```
def report(name, y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    print(f"{name:20s} | R²: {r2:.4f} | MAE: {mae:.4f}")
    return r2, mae

r2_lr, mae_lr = report("Linear Regression", y_test, y_pred_lr)
r2_rf, mae_rf = report("Random Forest", y_test, y_pred_rf)
```

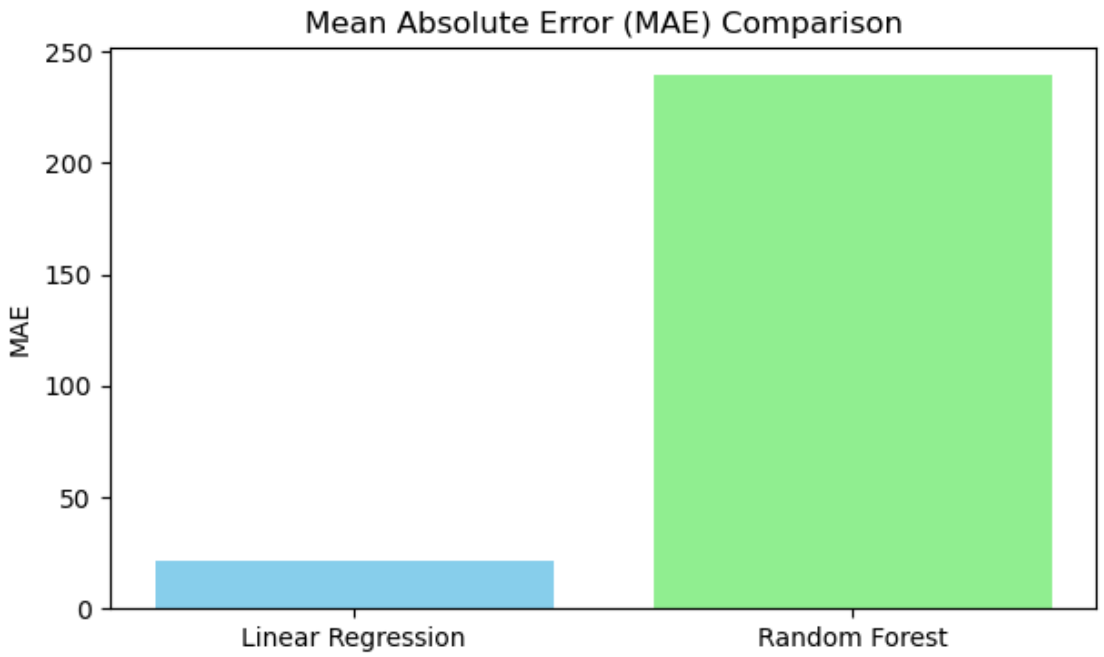
```
Linear Regression    | R²: 0.9590 | MAE: 21.7342
Random Forest        | R²: -2.8758 | MAE: 239.6037
```

[166]:
```
plt.figure(figsize=(12,5))
plt.plot(y_test.values, label="Actual Price", color='black', linewidth=2)
plt.plot(y_pred_lr, label="Linear Regression", alpha=0.7)
plt.plot(y_pred_rf, label="Random Forest", alpha=0.7)
plt.title("Actual vs Predicted Prices (Comparison)")
plt.xlabel("Time")
plt.ylabel("Price")
plt.legend()
plt.tight_layout()
plt.show()


models = ["Linear Regression", "Random Forest"]
r2_values = [r2_lr, r2_rf]
mae_values = [mae_lr, mae_rf]
```

```
fig, ax1 = plt.subplots(1, 1, figsize=(7,4))
ax1.bar(models, r2_values, color=['skyblue', 'lightgreen'])
ax1.set_title("R² Comparison")
ax1.set_ylabel("R² Score")
plt.show()

fig, ax2 = plt.subplots(1, 1, figsize=(7,4))
ax2.bar(models, mae_values, color=['skyblue', 'lightgreen'])
ax2.set_title("Mean Absolute Error (MAE) Comparison")
ax2.set_ylabel("MAE")
plt.show()
```






[167]:
```
from sklearn import datasets
```

[168]:
```
dfdata = sns.load_dataset("dowjones")
dfdata.head()
```

[168]:

| | Date | Price |
|---|---|---|
| 0 | 1914-12-01 | 55.00 |
| 1 | 1915-01-01 | 56.55 |
| 2 | 1915-02-01 | 56.00 |
| 3 | 1915-03-01 | 58.30 |
| 4 | 1915-04-01 | 66.45 |

[169]:
```
df = pd.DataFrame(dfdata.Date , columns = dfdata.Price)
```

[170]:
```
df.head()
```

[170]:
```
Empty DataFrame
Columns: [55.0, 56.55, 56.0, 58.3, 66.45, 65.95, 68.4, 71.85, 79.25, 85.5,
92.35, 94.35, 97.0, 94.7, 93.55, 93.3, 89.75, 90.15, 90.65, 88.45, 91.0, 97.45,
102.1, 107.9, 98.5, 97.15, 90.95, 94.65, 93.9, 93.35, 96.95, 92.9, 88.65, 83.6,
79.05, 71.4, 70.2, 76.6, 79.95, 78.05, 77.65, 81.05, 80.45, 81.85, 81.95, 82.5,
86.25, 84.0, 82.5, 81.65, 82.45, 86.55, 91.15, 99.4, 103.6, 109.7, 103.25,
108.2, 113.9, 111.6, 105.8, 105.9, 96.5, 97.95, 99.45, 91.1, 91.7, 90.7, 85.25,
```

```
86.5, 84.85, 79.3, 72.2, 74.75, 75.7, 75.05, 77.0, 76.7, 69.2, 68.6, 66.95,
69.35, 71.7, 75.7, 79.8, 80.6, 83.75, 87.2, 91.3, 93.95, 93.55, 95.0, 98.5,
99.2, 99.75, 95.75, 97.0, 98.2, 100.8, 103.9, …]
Index: []

[0 rows x 649 columns]
```

[171]:
```
df['Price'] = dfdata['Price']
```

[172]:
```
from sklearn.model_selection import train_test_split as split
```

[173]:
```
X_train , X_test , y_train , y_test = split(dfdata.Price , dfdata.Price ,␣
↪test_size = 0.2)
```

[174]:
```
from sklearn import svm
```

[179]:
```
import numpy as np

if not isinstance(X_train, np.ndarray):
    X_train = X_train.to_numpy()
if not isinstance(X_test, np.ndarray):
    X_test = X_test.to_numpy()
if not isinstance(y_train, np.ndarray):
    y_train = np.ravel(y_train)
if not isinstance(y_test, np.ndarray):
    y_test = np.ravel(y_test)


if X_train.ndim == 1:
    X_train = X_train.reshape(-1, 1)
if X_test.ndim == 1:
    X_test = X_test.reshape(-1, 1)
```

[180]:
```
X_train = np.asarray(X_train)
X_test = np.asarray(X_test)
y_train = np.asarray(y_train)
y_test = np.asarray(y_test)


if X_train.ndim == 1:
    X_train = X_train.reshape(-1, 1)
if X_test.ndim == 1:
    X_test = X_test.reshape(-1, 1)
y_train = y_train.flatten()
y_test = y_test.flatten()
```

[181]:
```
from sklearn.model_selection import GridSearchCV
```

[182]:
```
srv = GridSearchCV(svm.SVC(gamma = 'auto'),
                   {'C' :[1,10,20],
                    'kernel':['rbf' ,'linear']},
                    cv = 5,return_train_score = False)
```

[183]:
```
X = dfc[['ret_1','ma_5','vol_5']]
y = dfc['Price']
```

[184]:
```
from sklearn.svm import SVR
pipe = make_pipeline(StandardScaler(), SVR())

param_grid = {'svr__kernel': ['rbf', 'linear'],
              'svr__C': [0.1, 1, 10, 100],
              'svr__gamma': ['scale', 'auto']}

grid = GridSearchCV(pipe, param_grid, scoring='neg_mean_squared_error', cv=3,␣
 ↪n_jobs=-1, verbose=0)
grid.fit(X_train, y_train)
```

[184]:
```
GridSearchCV(cv=3,
             estimator=Pipeline(steps=[('standardscaler', StandardScaler()),
                                       ('svr', SVR())]),
             n_jobs=-1,
             param_grid={'svr__C': [0.1, 1, 10, 100],
                         'svr__gamma': ['scale', 'auto'],
                         'svr__kernel': ['rbf', 'linear']},
             scoring='neg_mean_squared_error')
```

[185]:
```
param_grid = {
    'kernel': ['linear', 'rbf'],
    'C': [0.1, 1, 10],
    'gamma': ['scale', 'auto']
}

grid_search = GridSearchCV(SVR(), param_grid, cv=5)
grid_search.fit(X_train, y_train)
```

[185]:
```
GridSearchCV(cv=5, estimator=SVR(),
             param_grid={'C': [0.1, 1, 10], 'gamma': ['scale', 'auto'],
                         'kernel': ['linear', 'rbf']})
```

[186]:
```
srv = GridSearchCV(model, param_grid, cv=3, scoring='r2', n_jobs=-1)
```

[189]:
```
from sklearn.svm import SVR

svr = SVR()
svr.fit(X_train, y_train)
```

```
y_pred_svr = svr.predict(X_test)
```

[190]:
```
param_grid = {
    'kernel': ['linear', 'rbf'],
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto']
}
```

[191]:
```
grid_search.cv_results_
```

[191]:
```
{'mean_fit_time': array([0.00108385, 0.00828385, 0.0006237 , 0.00573945,
0.0005024 ,
        0.00542922, 0.00045662, 0.00534458, 0.00071254, 0.00850444,
        0.00064778, 0.0095192 ]),
 'std_fit_time': array([3.41996472e-04, 1.75079654e-03, 1.02461754e-04,
4.07793719e-04,
        2.00188710e-05, 1.34500320e-04, 2.32989486e-05, 3.63514655e-04,
        1.90951856e-04, 9.57246140e-04, 1.38559230e-04, 1.50043790e-03]),
 'mean_score_time': array([0.00085921, 0.00567489, 0.00061202, 0.0043458 ,
0.00049405,
        0.0040956 , 0.00051703, 0.00411634, 0.00066457, 0.00485358,
        0.00068817, 0.00463767]),
 'std_score_time': array([1.34080080e-04, 1.00551961e-03, 6.66561383e-05,
2.93400643e-04,
        5.62465761e-06, 1.43861652e-04, 7.74348310e-05, 3.37990150e-04,
        1.31205156e-04, 4.34260409e-04, 9.34912912e-05, 1.88398319e-04]),
 'param_C': masked_array(data=[0.1, 0.1, 0.1, 0.1, 1.0, 1.0, 1.0, 1.0, 10.0,
10.0,
                    10.0, 10.0],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value=1e+20),
 'param_gamma': masked_array(data=['scale', 'scale', 'auto', 'auto', 'scale',
'scale',
                    'auto', 'auto', 'scale', 'scale', 'auto', 'auto'],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_kernel': masked_array(data=['linear', 'rbf', 'linear', 'rbf', 'linear',
'rbf',
                    'linear', 'rbf', 'linear', 'rbf', 'linear', 'rbf'],
              mask=[False, False, False, False, False, False, False, False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'},
```

```
  {'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'},
  {'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'},
  {'C': 0.1, 'gamma': 'auto', 'kernel': 'rbf'},
  {'C': 1, 'gamma': 'scale', 'kernel': 'linear'},
  {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'},
  {'C': 1, 'gamma': 'auto', 'kernel': 'linear'},
  {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'},
  {'C': 10, 'gamma': 'scale', 'kernel': 'linear'},
  {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'},
  {'C': 10, 'gamma': 'auto', 'kernel': 'linear'},
  {'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}],
 'split0_test_score': array([ 0.99999993, -0.27503558,  0.99999993, -0.34674163,
0.99999993,
        0.27957281,  0.99999993, -0.33860568,  0.99999993,  0.98978917,
        0.99999993, -0.26658791]),
 'split1_test_score': array([ 0.99999993, -0.22531024,  0.99999993, -0.30073595,
0.99999993,
        0.33822966,  0.99999993, -0.30149746,  0.99999993,  0.99187353,
        0.99999993, -0.252192  ]),
 'split2_test_score': array([ 0.99999992, -0.09035798,  0.99999992, -0.15250005,
0.99999992,
        0.37965392,  0.99999992, -0.14892305,  0.99999992,  0.98028337,
        0.99999992, -0.09434162]),
 'split3_test_score': array([ 0.99999987,  0.0057193 ,  0.99999987, -0.06809463,
0.99999987,
        0.54136339,  0.99999987, -0.06465951,  0.99999987,  0.99692366,
        0.99999987, -0.01147313]),
 'split4_test_score': array([ 0.99999993, -0.20533687,  0.99999993, -0.27301928,
0.99999993,
        0.30530087,  0.99999993, -0.27083871,  0.99999993,  0.98153106,
        0.99999993, -0.22806874]),
 'mean_test_score': array([ 0.99999991, -0.15806427,  0.99999991, -0.22821831,
0.99999991,
        0.36882413,  0.99999991, -0.22490488,  0.99999991,  0.98808016,
        0.99999991, -0.17053268]),
 'std_test_score': array([2.44190805e-08, 1.01831436e-01, 2.44190805e-08,
1.02651409e-01,
       2.44190805e-08, 9.25463174e-02, 2.44190805e-08, 1.02328910e-01,
       2.44190805e-08, 6.31186984e-03, 2.44190805e-08, 1.00309786e-01]),
 'rank_test_score': array([ 1,  9,  1, 12,  1,  8,  1, 11,  1,  7,  1, 10],
dtype=int32)}
```

[192]:
```
srv = GridSearchCV(model, param_grid, cv=3, scoring='r2', n_jobs=-1)


srv.fit(X_train, y_train)
```

```
best = pd.DataFrame(srv.cv_results_)
print(srv.best_params_)
```

```
{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
```

[193]:
```
srv.best_score_
```

[193]:
```
0.9999999194621404
```