4-sales-prediction-using-python

July 8, 2024

[ ]:
```
TASK 4 - SALES PREDICTION USING PYTHON

Sales prediction involves forecasting the amount of a product tha customers␣
 ↪will purchase,
taking into account various factors such as
"advertising expenditure, target audience segmentation, and advertising␣
 ↪platform selection".

In businesses that offer products or services, the role of a Data Scientist is␣
 ↪crucial for predicting future sales.
They utilize machine learning techniques in Python to analyze and interpret␣
 ↪data,
allowing them to make informed decisions regarding advertising costs.
By leveraging these predictions, businesses can optimize their advertising␣
 ↪strategies and maximize sales potential.
Let's embark on the journey of sales prediction using machine learning in␣
 ↪Python.
```

[ ]:
```

```

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import plotly.io as plio
plio.templates
import plotly.express as px
import plotly.graph_objects as go
from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, r2_score

from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from xgboost import XGBRegressor
```

```python
import joblib

from warnings import filterwarnings
filterwarnings(action='ignore')
```

[2]:
```python
data = pd.read_csv(r"C:\Users\divya\OneDrive\Documents\CodSoft␣
↪Internship\advertising.csv")
```

[3]:
```python
data
```

[3]:
```
        TV  Radio  Newspaper  Sales
0    230.1   37.8       69.2   22.1
1     44.5   39.3       45.1   10.4
2     17.2   45.9       69.3   12.0
3    151.5   41.3       58.5   16.5
4    180.8   10.8       58.4   17.9
..     …      …         …     …
195   38.2    3.7       13.8    7.6
196   94.2    4.9        8.1   14.0
197  177.0    9.3        6.4   14.8
198  283.6   42.0       66.2   25.5
199  232.1    8.6        8.7   18.4

[200 rows x 4 columns]
```

[4]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

[6]:
```python
data.describe()
```

[6]:
```
               TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500   23.264000   30.554000   15.130500
std     85.854236   14.846809   21.778621    5.283892
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   11.000000
```

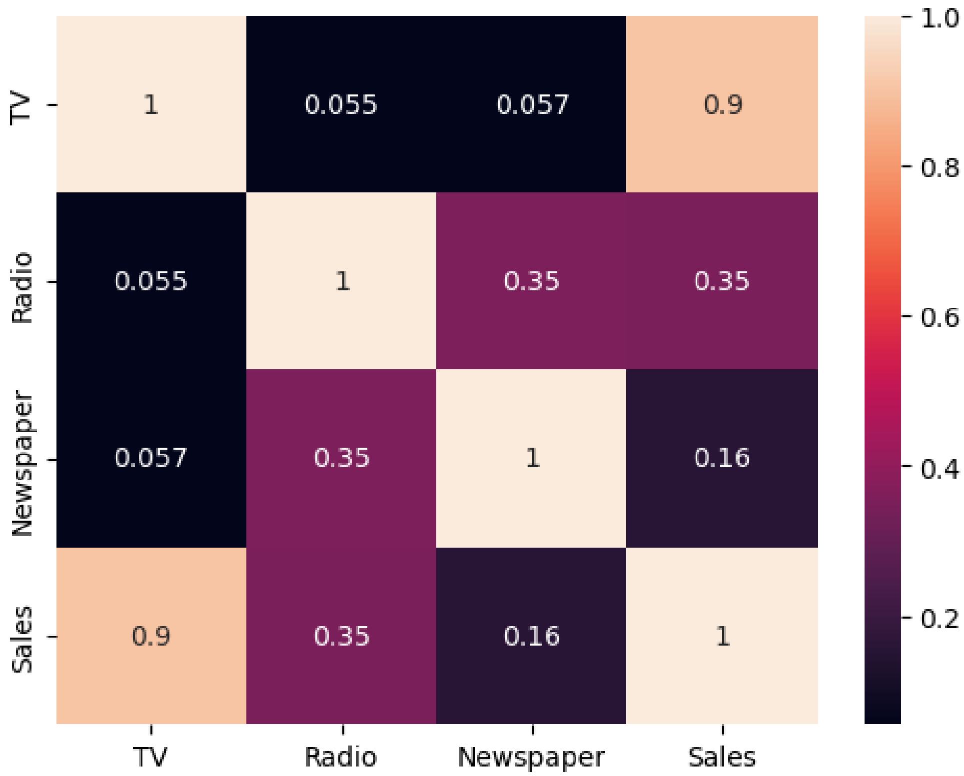
```
50%    149.750000   22.900000   25.750000   16.000000
75%    218.825000   36.525000   45.100000   19.050000
max    296.400000   49.600000  114.000000   27.000000
```

[7]:
```
data.duplicated().sum()
```

[7]: 0

[8]:
```
sns.heatmap(data.corr(),annot=True)
```

[8]: <Axes: >



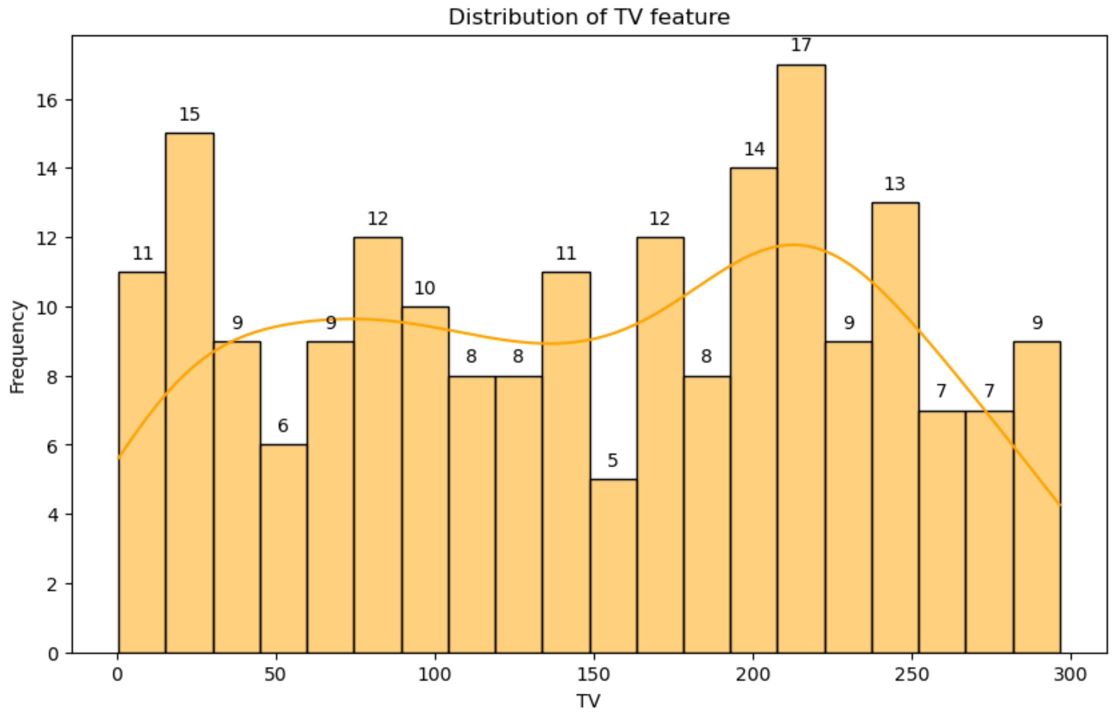
[9]:
```
fig = go.Figure(data=go.Scatter(x=data['TV'], y=data['Sales'], mode='markers',␣
    ↪marker=dict(color='orange', size=8)))
fig.update_layout(
    title="Scatter Plot of TV(Feature) vs Sales(Target)",
    xaxis_title="TV",
    yaxis_title="Sales"
)
fig.show()
```

[10]:
```
fig = go.Figure(data=go.Scatter(x=data['Radio'], y=data['Sales'],␣
↪mode='markers', marker=dict(color='red', size=8)))
fig.update_layout(
    title="Scatter Plot of Radio(Feature) vs Sales(Target)",
    xaxis_title="Radio",
    yaxis_title="Sales"
)
fig.show()
```

[11]:
```
fig = go.Figure(data=go.Scatter(x=data['Newspaper'], y=data['Sales'],␣
↪mode='markers', marker=dict(color='blue', size=8)))
fig.update_layout(
    title="Scatter Plot of Newspaper(Feature) vs Sales(Target)",
    xaxis_title="Newspaper",
    yaxis_title="Sales"
)
fig.show()
```

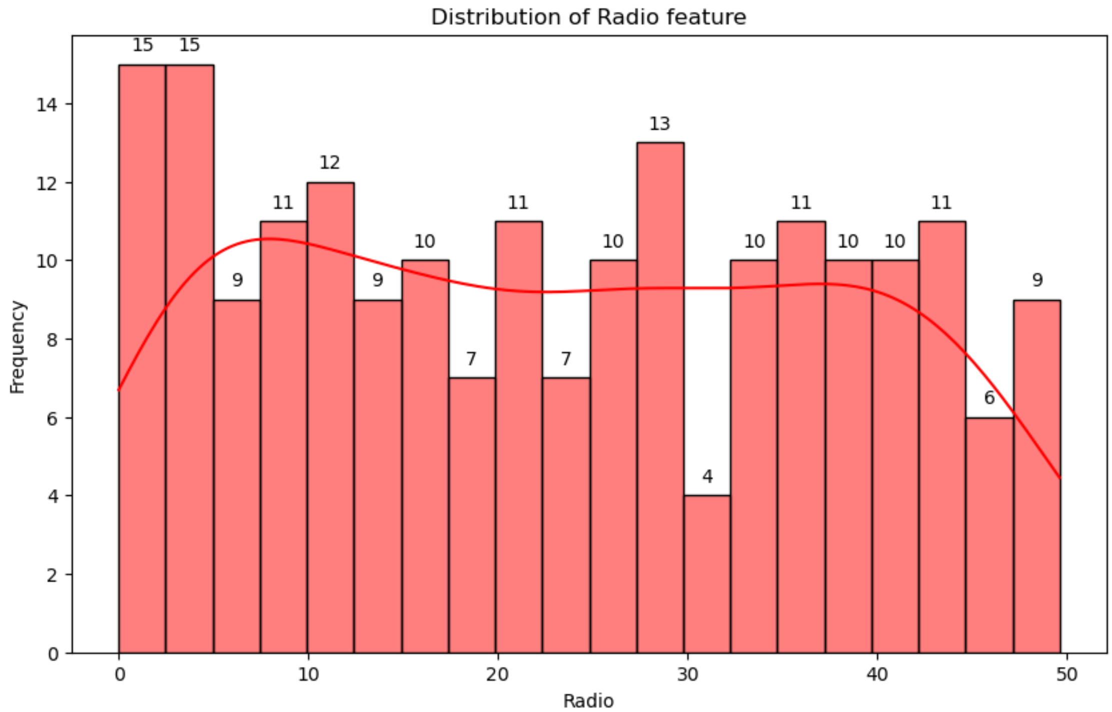
[ ]:
```
1. DISTRIBUTION OF TV FEATURE
```

[12]:
```
plt.figure(figsize=(10, 6))
ax = sns.histplot(data['TV'], bins=20, kde=True, color='orange')
plt.xlabel('TV')
plt.ylabel('Frequency')
plt.title('Distribution of TV feature')
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
↪xytext=(0, 10),
                textcoords='offset points')
plt.show()
```



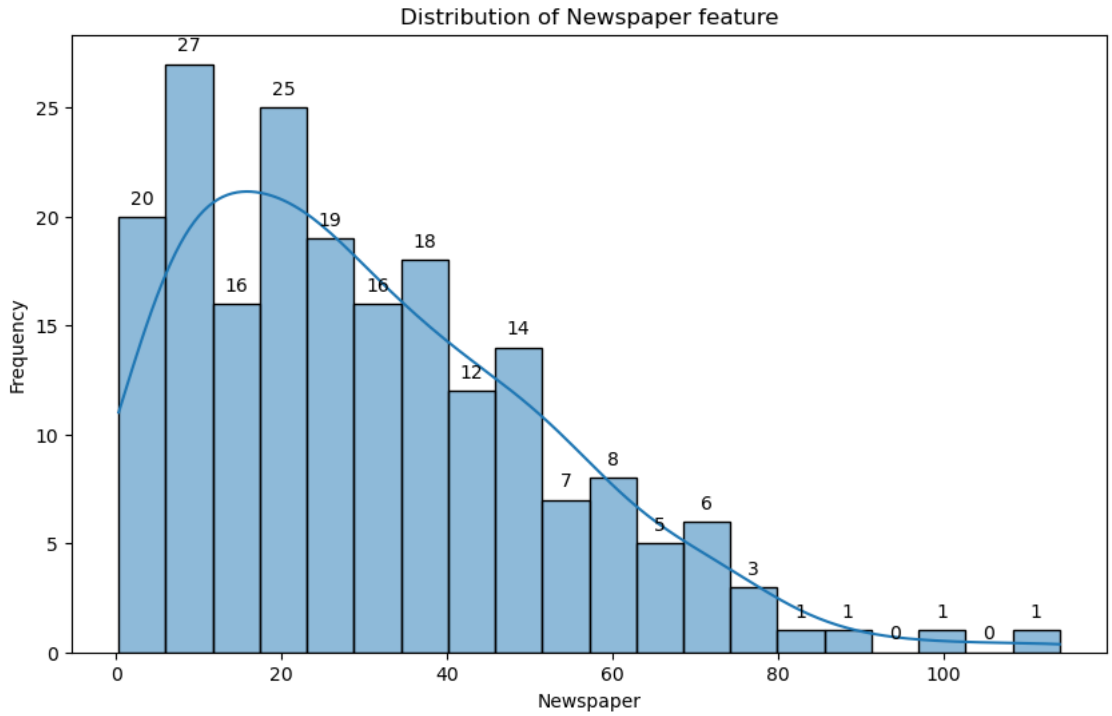

```
[ ]: 2. DISTRIBUTION OF RADIO  FEATURE
```

```
[13]: plt.figure(figsize=(10, 6))
ax = sns.histplot(data['Radio'], bins=20, kde=True, color='red')
plt.xlabel('Radio')
plt.ylabel('Frequency')
plt.title('Distribution of Radio feature')
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 10),
                textcoords='offset points')
plt.show()
```



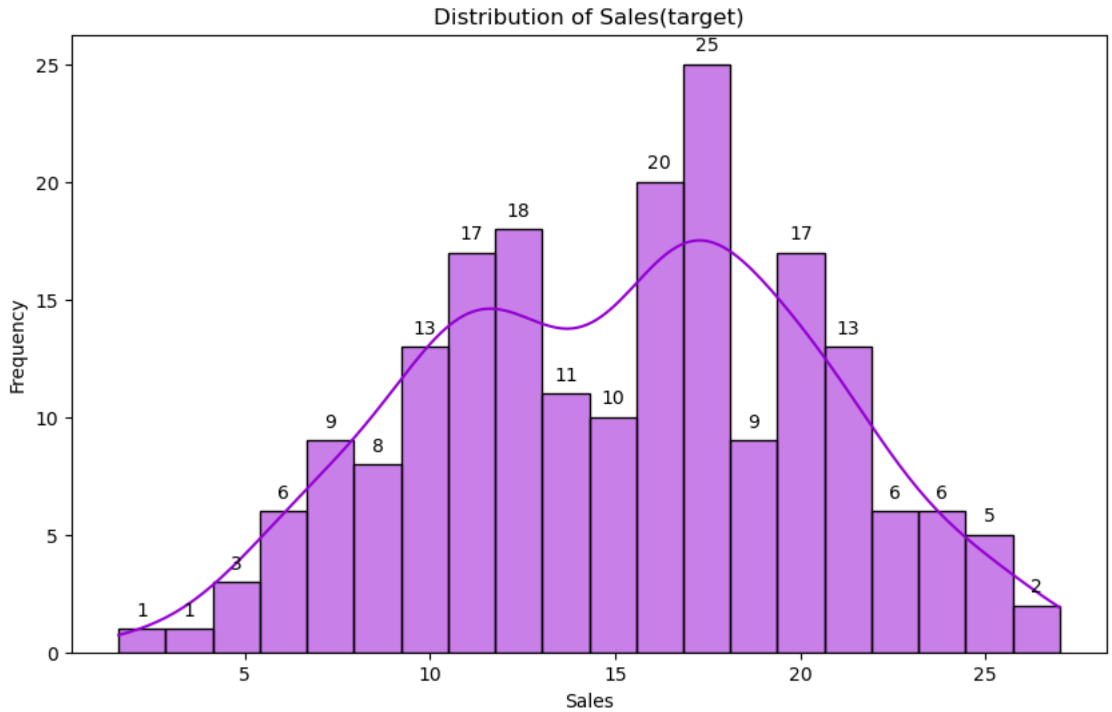
[ ]: 3. DISTRIBUTION OF NEWSPAPER FEATURE

[14]:
```
plt.figure(figsize=(10, 6))
ax = sns.histplot(data['Newspaper'], bins=20, kde=True)
plt.xlabel('Newspaper')
plt.ylabel('Frequency')
plt.title('Distribution of Newspaper feature')
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 10),
                textcoords='offset points')
plt.show()
```



[ ]: 4. DISTRIBUTION OF SALES

[15]:
```
plt.figure(figsize=(10, 6))
ax = sns.histplot(data['Sales'], bins=20, kde=True, color='darkviolet')
plt.xlabel('Sales')
plt.ylabel('Frequency')
plt.title('Distribution of Sales(target)')
for p in ax.patches:
    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.
 ↪get_height()),
                ha='center', va='center', fontsize=10, color='black',␣
 ↪xytext=(0, 10),
                textcoords='offset points')
```



[16]:
```
x = data.iloc[:,:3]
y = data.iloc[:,3:]
```

[17]:
```
x
```

[17]:
```
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     …      …         …
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
```

[18]:
```
y
```

[18]:
```
     Sales
0     22.1
```

```
1      10.4
2      12.0
3      16.5
4      17.9
..      …
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4

[200 rows x 1 columns]
```

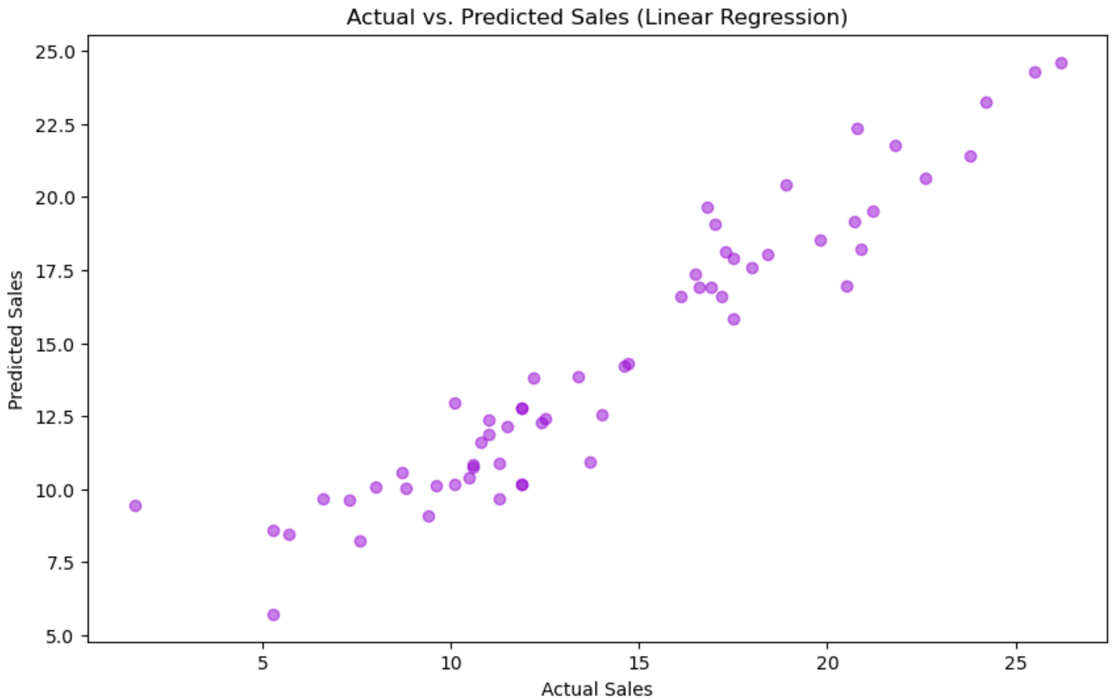
[19]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

[20]:
```
model_1 = LinearRegression()
model_1.fit(x_train, y_train)
y_pred_1 = model_1.predict(x_test)
```

[21]:
```
mse = mean_squared_error(y_test, y_pred_1)
r2 = r2_score(y_test, y_pred_1)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 3.4038325522795776
R-Squared score is : 0.8875641116008756
```

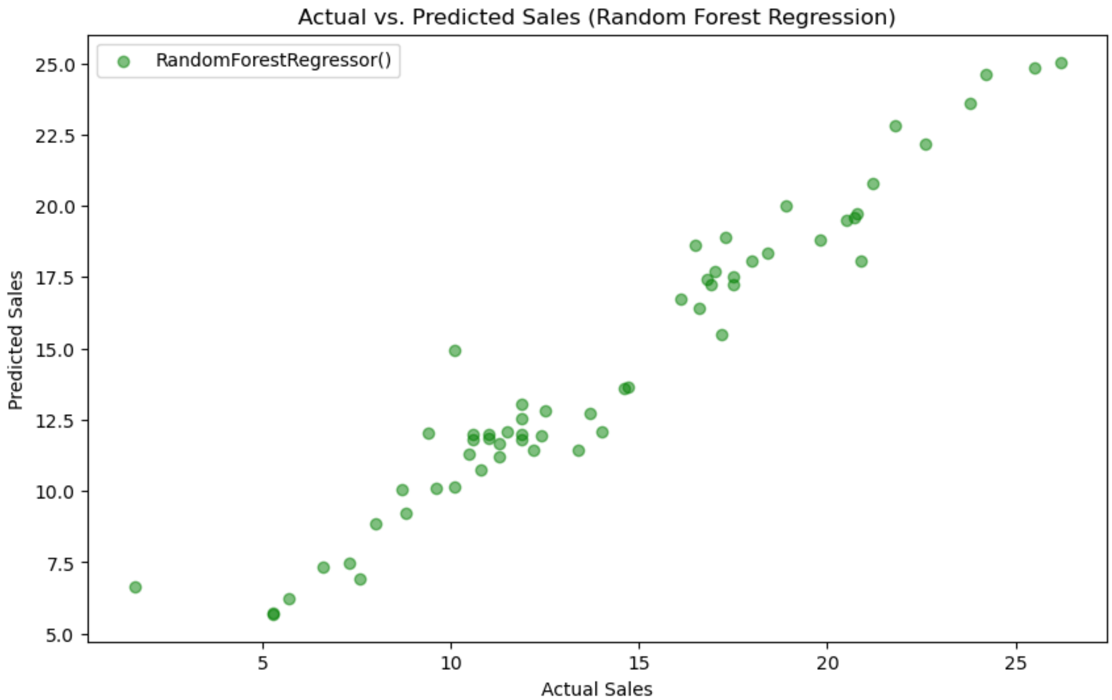
[22]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_1, alpha=0.5, color='darkviolet')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (Linear Regression)')
plt.show()
```



[23]:
```
model_2 = RandomForestRegressor()
model_2.fit(x_train, y_train)
y_pred_2 = model_2.predict(x_test)
mse = mean_squared_error(y_test, y_pred_2)
r2 = r2_score(y_test, y_pred_2)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 1.8022122666666631
R-Squared score is : 0.940469064158059
```

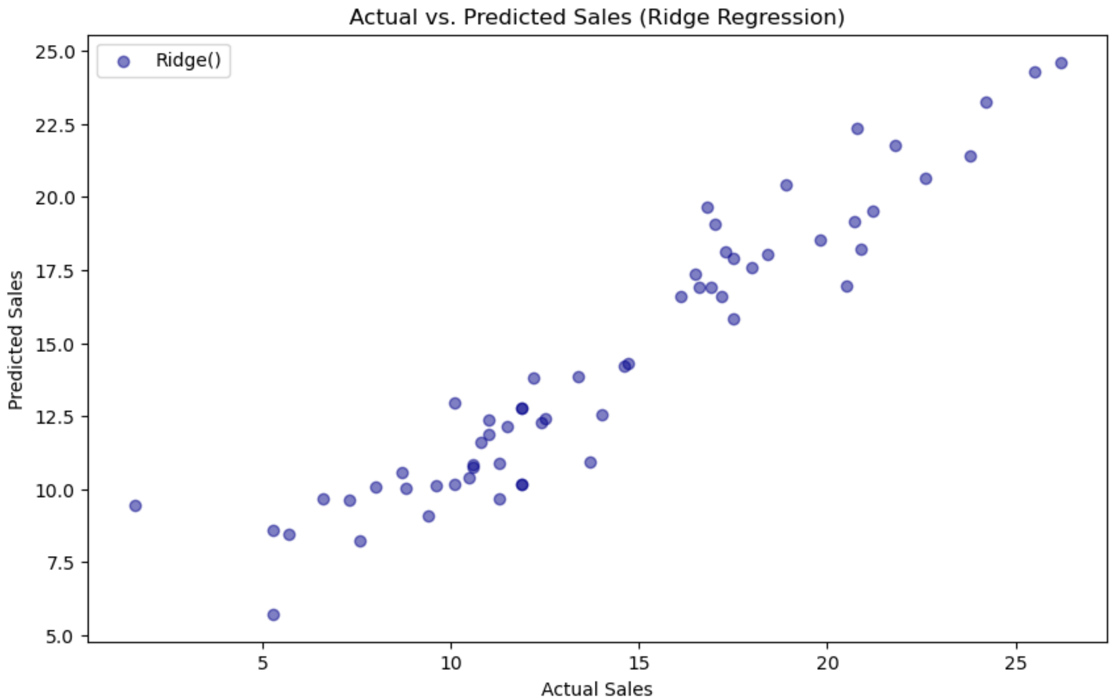
[24]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_2, label=model_2, alpha=0.5, color='green')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (Random Forest Regression)')
plt.legend()
plt.show()
```



[25]:
```
model_3 = Ridge()
model_3.fit(x_train, y_train)
y_pred_3 = model_3.predict(x_test)
mse = mean_squared_error(y_test, y_pred_3)
r2 = r2_score(y_test, y_pred_3)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 3.4038169687278437
R-Squared score is : 0.8875646263590068
```

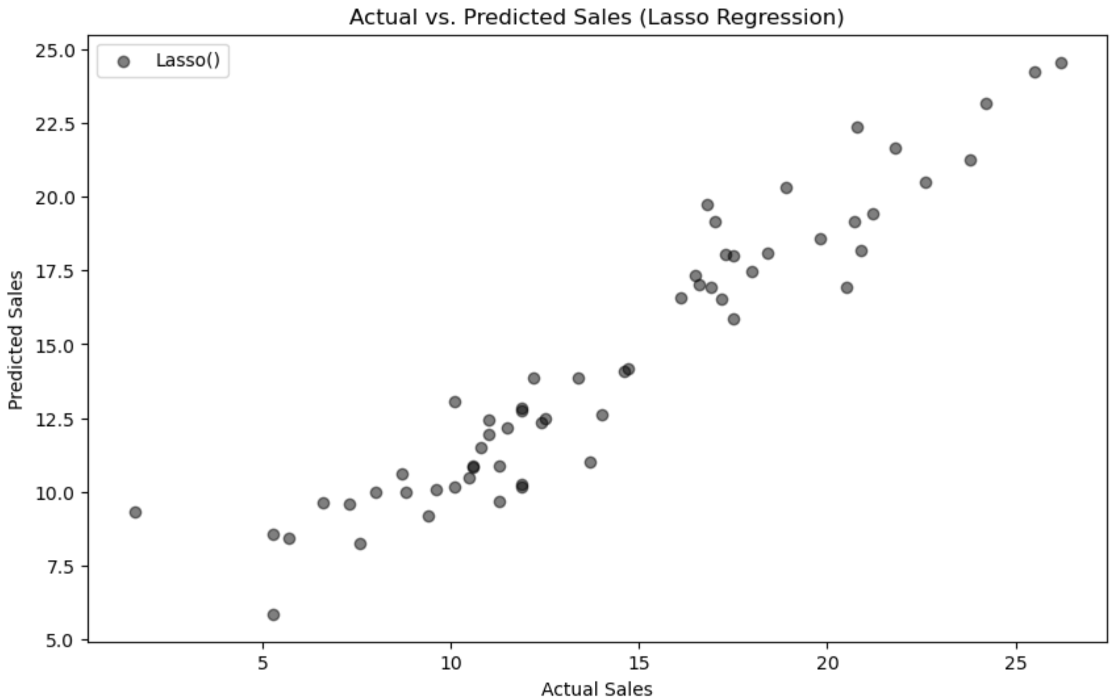
[26]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_3, label=model_3, alpha=0.5, color='darkblue')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (Ridge Regression)')
plt.legend()
plt.show()
```



[27]:
```
model_4 = Lasso()
model_4.fit(x_train, y_train)
y_pred_4 = model_4.predict(x_test)
mse = mean_squared_error(y_test, y_pred_4)
r2 = r2_score(y_test, y_pred_4)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 3.3925415015877234
R-Squared score is : 0.887937079217819
```

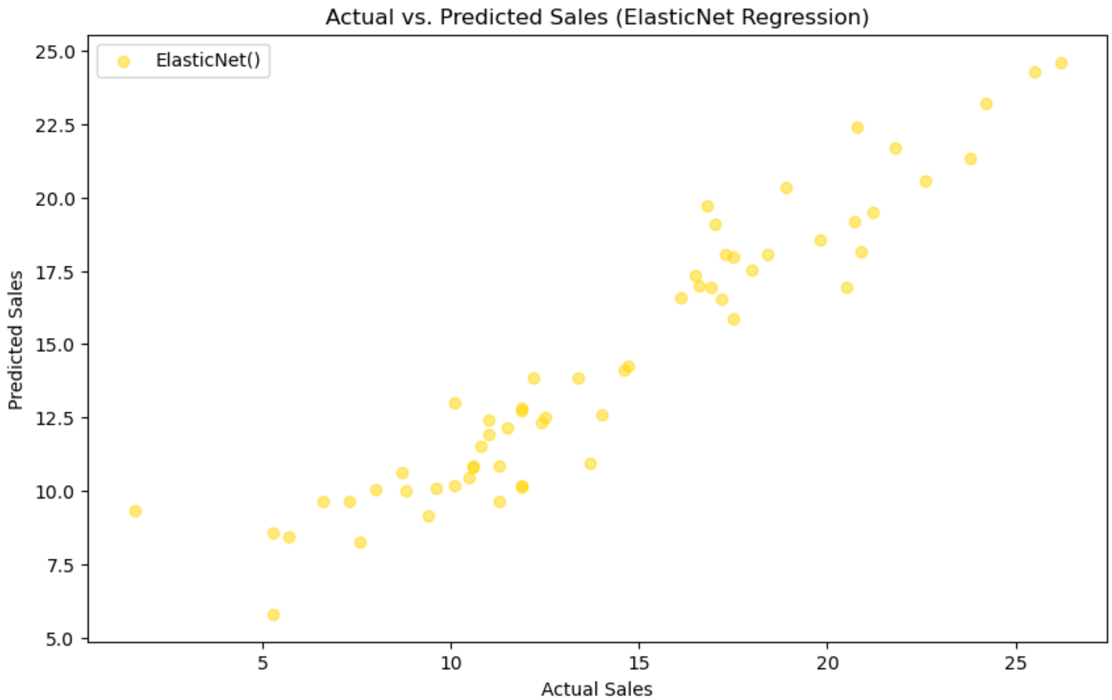
[28]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_4, label=model_4, alpha=0.5, color='black')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (Lasso Regression)')
plt.legend()
plt.show()
```



[29]:
```
model_5 = ElasticNet()
model_5.fit(x_train, y_train)
y_pred_5 = model_5.predict(x_test)
mse = mean_squared_error(y_test, y_pred_5)
r2 = r2_score(y_test, y_pred_5)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 3.390505094047334
R-Squared score is : 0.8880043461257616
```

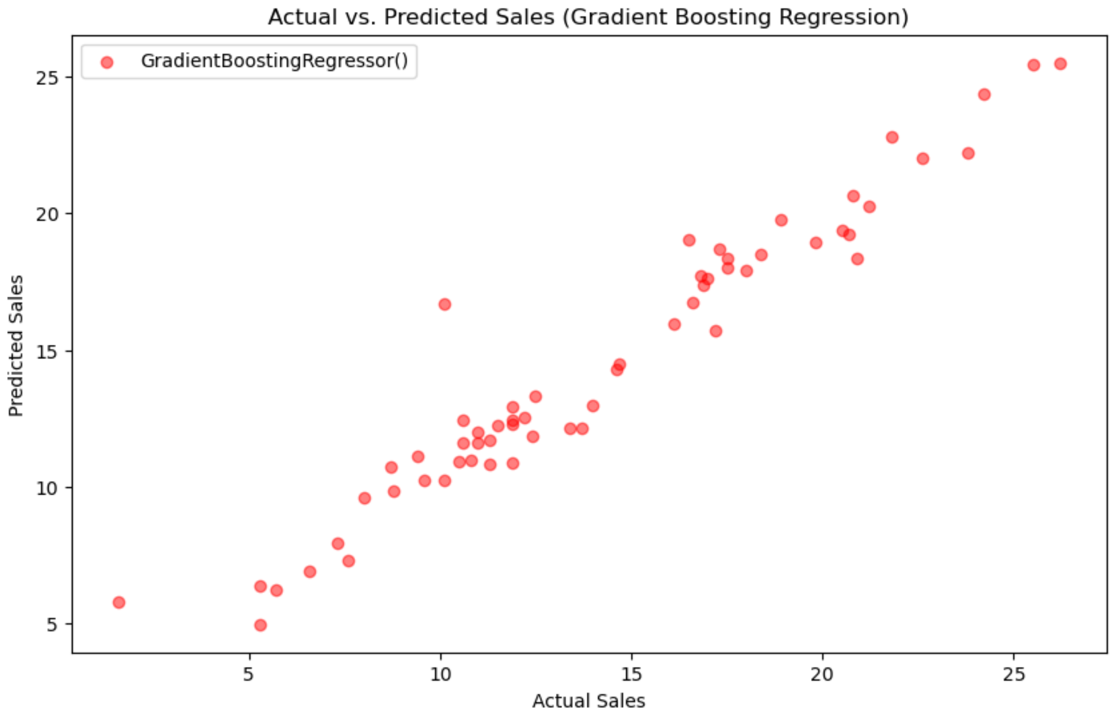
[30]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_5, label=model_5, alpha=0.5, color='gold')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (ElasticNet Regression)')
plt.legend()
plt.show()
```



[31]:
```
model_6 = GradientBoostingRegressor()
model_6.fit(x_train, y_train)
y_pred_6 = model_6.predict(x_test)
mse = mean_squared_error(y_test, y_pred_6)
r2 = r2_score(y_test, y_pred_6)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 2.0098563088322723
R-Squared score is : 0.9336101361722977
```

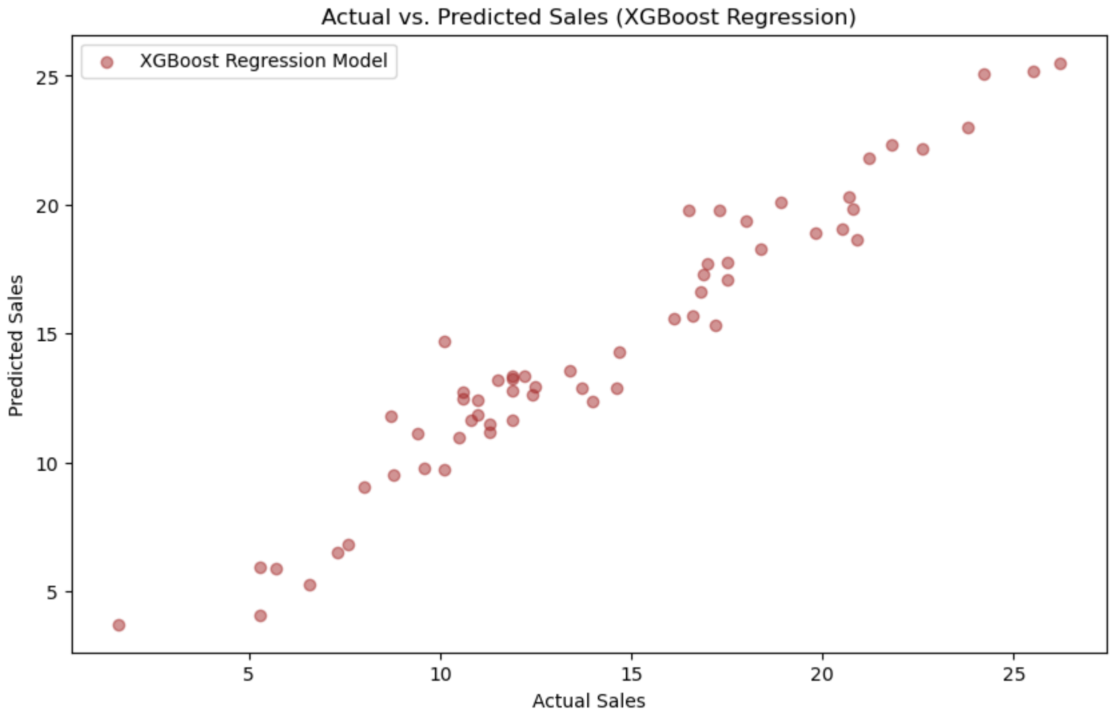
[32]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_6, label=model_6, alpha=0.5, color='red')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (Gradient Boosting Regression)')
plt.legend()
plt.show()
```



[33]:
```
model_7 = XGBRegressor()
model_7.fit(x_train, y_train)
y_pred_7 = model_7.predict(x_test)
mse = mean_squared_error(y_test, y_pred_7)
r2 = r2_score(y_test, y_pred_7)
print("Mean Square Error is :", mse)
print("R-Squared score is :",r2)
```

```
Mean Square Error is : 1.82916503673819
R-Squared score is : 0.9395787563649274
```

[34]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_7, label='XGBoost Regression Model', alpha=0.5,␣
 ↪color='brown')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Actual vs. Predicted Sales (XGBoost Regression)')
plt.legend()
plt.show()
```



[35]:
```
model_r2_scores = {
     "Linear Regression Model":  r2_score(y_test, y_pred_1),

     "Random Forest Regression Model":  r2_score(y_test, y_pred_2),

     "Ridge Regression Model":  r2_score(y_test, y_pred_3),

     "Lasso Regression Model":  r2_score(y_test, y_pred_4),

     "ElasticNet Regression Model":  r2_score(y_test, y_pred_5),

     "Gradient Boosting Regression Model": r2_score(y_test, y_pred_6),

     "XGBoost Regression Model":  r2_score(y_test, y_pred_7)
}
model_r2_scores
```

[35]:
```
{'Linear Regression Model': 0.8875641116008756,
 'Random Forest Regression Model': 0.940469064158059,
 'Ridge Regression Model': 0.8875646263590068,
 'Lasso Regression Model': 0.887937079217819,
 'ElasticNet Regression Model': 0.8880043461257616,
 'Gradient Boosting Regression Model': 0.9336101361722977,
 'XGBoost Regression Model': 0.9395787563649274}
```

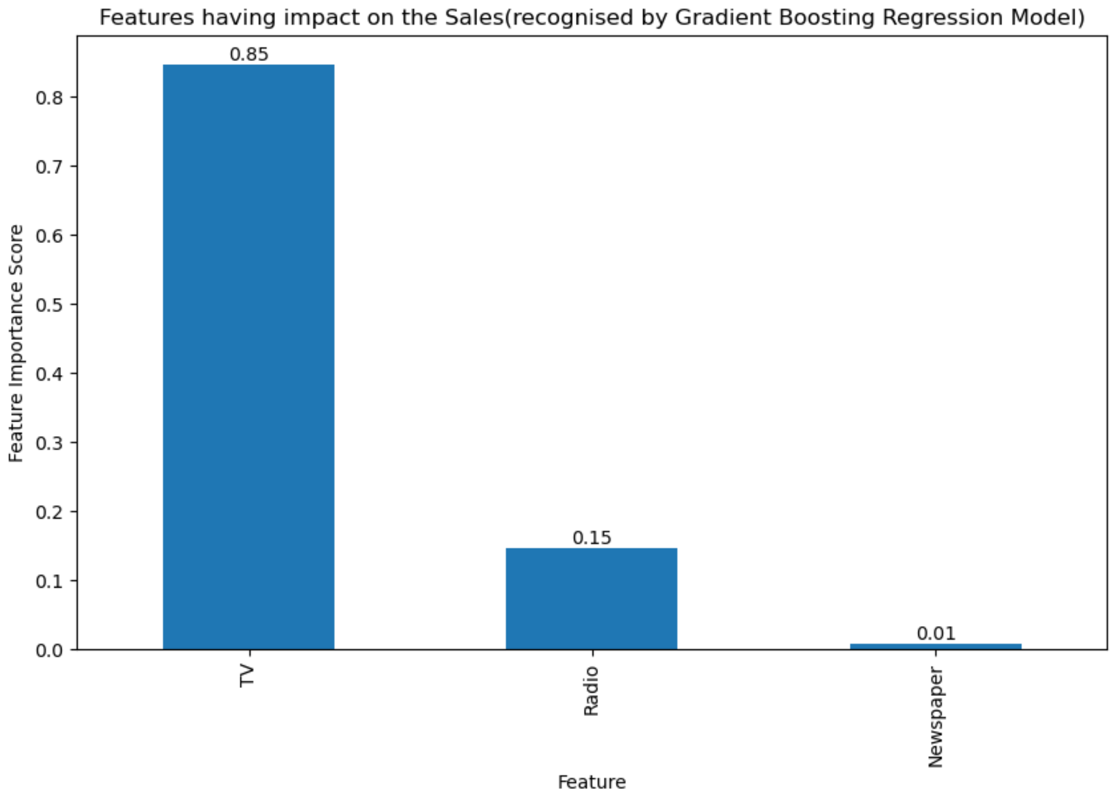
```
[36]: best_model_name = max(model_r2_scores, key=model_r2_scores.get)
best_r2_score = model_r2_scores[best_model_name]

print(f"Best Performing Model is {best_model_name} with an R^2 score of␣
↪{best_r2_score}")
```

```
Best Performing Model is Random Forest Regression Model with an R^2 score of
0.940469064158059
```

```
[37]: final_model = model_6
joblib.dump(final_model, 'gradient_boosting_model.pkl')
```

```
[37]: ['gradient_boosting_model.pkl']
```

```
[38]: feature_importances = pd.Series(final_model.feature_importances_, index=x.
↪columns)
plt.figure(figsize=(10, 6))
features = feature_importances
features.plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Feature Importance Score')
plt.title('Features having impact on the Sales(recognised by Gradient Boosting␣
↪Regression Model)')
for index, value in enumerate(features):
    plt.text(index, value, f'{value:.2f}', ha='center', va='bottom')
plt.show()
```


Features having impact on the Sales(recognised by Gradient Boosting Regression Model)
0.85
0.15
0.01
Feature Importance Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
TV
Radio
Newspaper
Feature

[ ]: