Samuel Castro

```
Project Milestone 1: Data Selection and EDA
```

Business Problem: The first step of this project was to identify a use for data mining in a business setting. I have settled on advertising expenditures vs sales. I have always enjoyed and appreciated regression models so I plan on building a regression model to predict the efficiency of different advertising channels. Every company contains some sort of advertising, and these different channels of advertising need to be looked at and utilized correctly to maximize sales. By looking at sales data, and advertising data one can build a regression model to predict how much sales each channel will generate based on how much was spent on advertising. The data I will be looking at is going to be advertising expenditures from three different channels. These different channels are, "Tv advertising", "Radio Advertising", and "Newspaper Advertising". Any company can use a multiple linear regression model to estimate a relationship between advertising spending and sales. The goal of this project is to show the potential of each respective channel of advertising. The model needs to be trained from historical data. By creating a model the company can then optimize the spending across the channels to maximize its revenue stream. They can also use the model to monitor the effectiveness of their advertising spending over time and adjust their strategies accordingly. Additionally, the regression model could be used to forecast revenue based on different advertising spending scenarios, enabling the company to make data-driven decisions about its advertising budget allocation. Analyzing advertising data is essential for any company because this can reveal underlying issues in a revenue stream. For example, newspaper and radio ads are not used as often as Facebook ads or e-commerce ads. So why do some companies continue advertising through newspapers or radio? By looking at the data we can see that newspapers and radio are still being watched by the consumer. We cannot make decisions without looking at data.

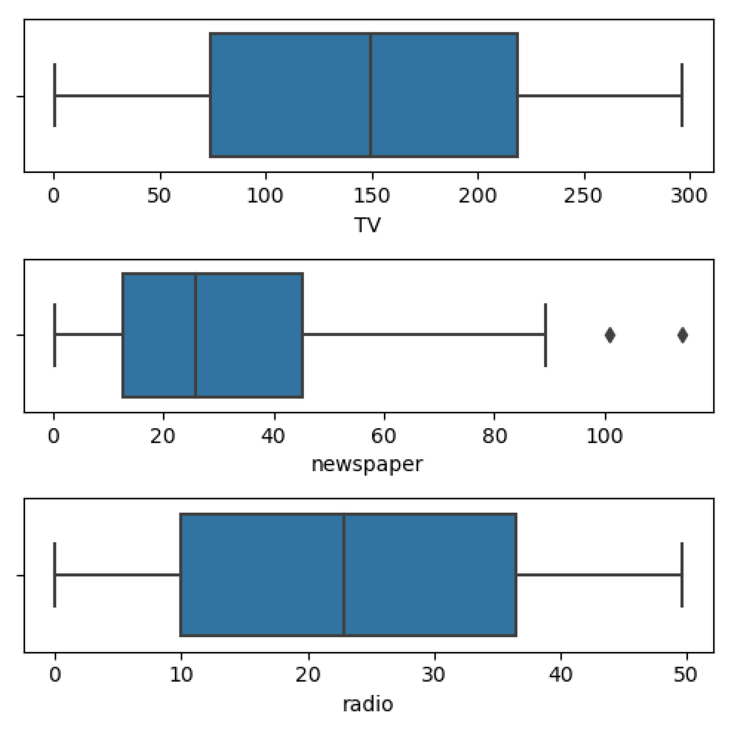
In [1]:
```python
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
df = pd.DataFrame(pd.read_csv("Desktop/DSC 550/Advertising Dataset.csv"))
df.head()
```

Out[2]:

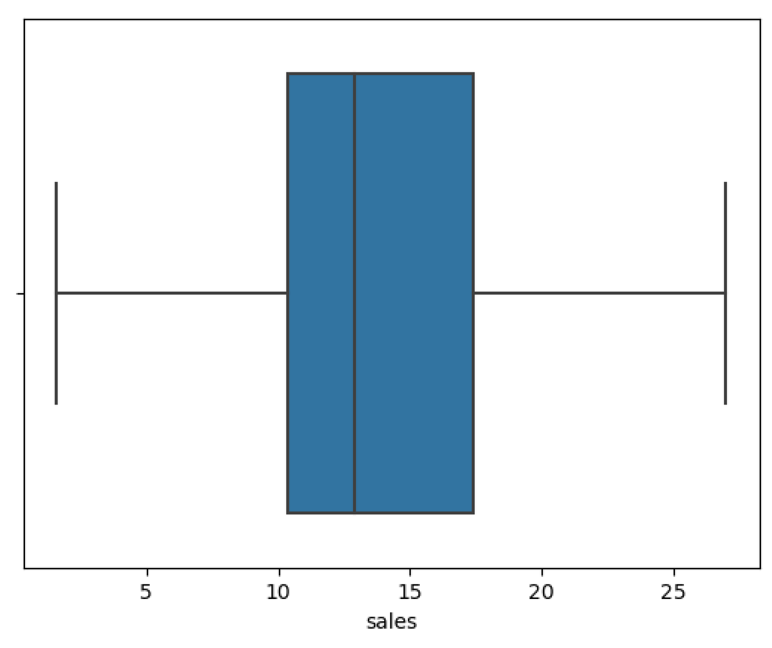
	S.no	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [3]:
```python
# Outlier Analysis
fig, axs = plt.subplots(3, figsize = (5,5))
plt1 = sns.boxplot(df['TV'], ax = axs[0])
plt2 = sns.boxplot(df['newspaper'], ax = axs[1])
plt3 = sns.boxplot(df['radio'], ax = axs[2])
plt.tight_layout()
```

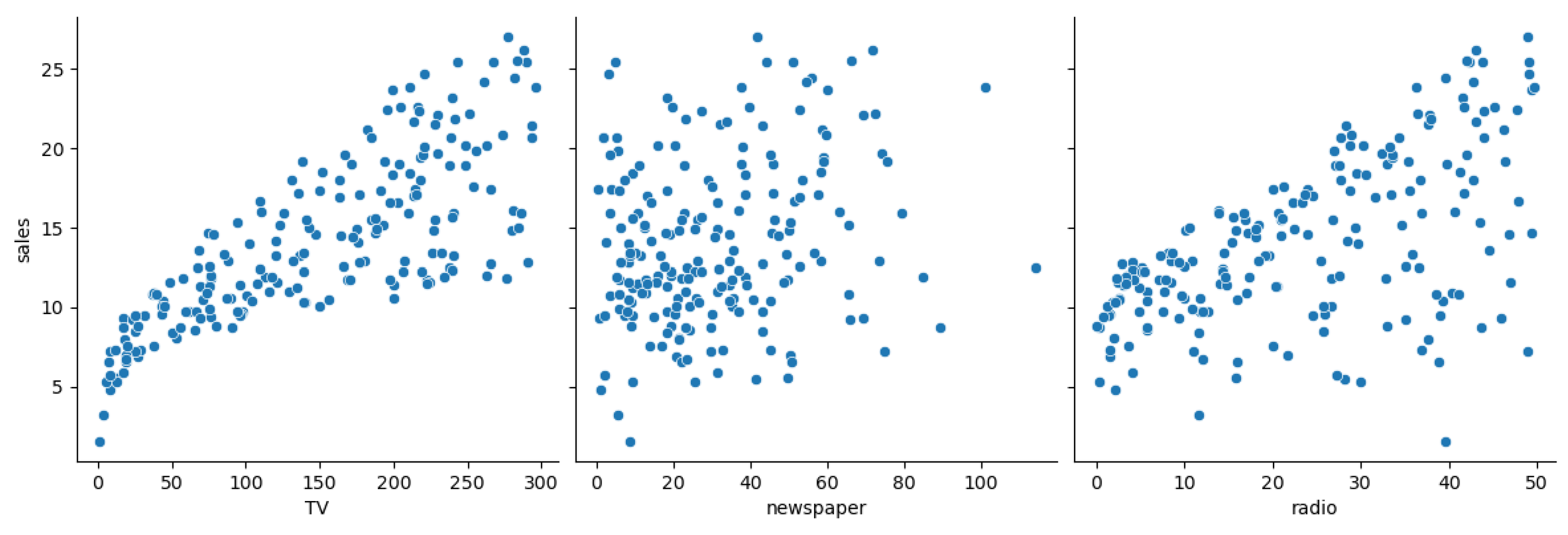


As you can see above, there are no outliers that are a threat to this model. In the newspaper boxplot there are some outliers but not many.

In [4]:
```python
sns.boxplot(df['sales'])
plt.show()
```



In [5]:
```python
sns.pairplot(df, x_vars=['TV', 'newspaper', 'radio'], y_vars='sales', height=4, aspect=1, kind='scatter')
plt.show()
```



As you can see above, the most correlated variable with sales seems to be TV. The other variables have very low expenditures in advertising but seem to be vaguely correlated with sales.

The conclusion on my very brief EDA is that each variable is correlated with sales in their respective ways. Tv seems to be the most correalted but our goal is to see which advertising stream generates the most revenue and which stream has the potential to create the most revenue.

```
Samuel Castro Project Mileston #2
```

Now that you have created your idea, located data, and have started your graphical analysis, you will move on to the data preparation process of your project. After completing Milestone 2, your data should be ready for the model building/evaluation phase.

In [6]:
```python
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

I know that this data is a very simple dataset and I know that I can produce a good example of building a model with this data.

In [7]:
```python
df = pd.DataFrame(pd.read_csv("Desktop/DSC 550/Advertising Dataset.csv"))
df.head()
```

Out[7]:

	S.no	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [8]:
```python
# check for missing values
df.isnull().sum()
# There are no missing values in the dataset, so we don't need to deal with them.
```

Out[8]:
```
S.no         0
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [9]:
```python
# drop the 'S.no' column
# The "S.no" column is not useful for model building as it just represents the row number, which is not a predictor # variable for sales. Therefore, we can drop it.
df = df.drop('S.no', axis=1)
df.head()
```

Out[9]:

	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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [10]:
```python
# Perform any data extraction/selection steps.
# There are no data extraction or selection steps to perform for this dataset.
```

In [11]:
```python
# Transform features if necessary.
# There are no features that need to be transformed for this dataset.
```

In [12]:
```python
# Engineer new useful features.
# There are no new features that need to be engineered for this dataset.
```

In [13]:
```python
# Create dummy variables if necessary.
# There are no categorical variables in this dataset that need to be converted to dummy variables.
# Therefore, we don't need to create any dummy variables.
```

Samuel Castro: Term Project Milestone 3: Model Building and Evaluation

In [14]:
```python
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [15]:
```python
df = pd.DataFrame(pd.read_csv("Desktop/DSC 550/Advertising Dataset.csv"))
df.head()
```

Out[15]:

	S.no	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [16]:
```python
# drop the 'S.no' column
# The "S.no" column is not useful for model building as it just represents the row number, which is not a predictor
# variable for sales. Therefore, we can drop it.
df = df.drop('S.no', axis=1)
df.head()
```

Out[16]:

	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	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

In [17]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, mean_squared_log_error
from sklearn.tree import DecisionTreeRegressor

# Split the data into input (advertising) and output (sales) variables

X = df[['TV', 'radio', 'newspaper']]
y = df['sales']

# Split the data into training and testing sets

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

# Train the linear regression model

model1 = LinearRegression()
model1.fit(X_train, y_train)

# Predict the sales values for the test set

y_pred1 = model1.predict(X_test)

# Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred1)
print('Mean Squared Error:', mse)

# Calculate Mean Absolute Error (MAE)

mae = mean_absolute_error(y_test, y_pred1)
print('Mean Absolute Error (MAE):', mae)

# Calculate R-squared (R2) Score

r2 = r2_score(y_test, y_pred1)
print('R-squared (R2) Score:', r2)

# Calculate Adjusted R-squared Score

n = X_test.shape[0] # Number of samples
p = X_test.shape[1] # Number of predictors
adjusted_r2 = 1 - ((1 - r2) * (n - 1)) / (n - p - 1)
print('Adjusted R-squared Score:', adjusted_r2)

# Calculate Mean Squared Logarithmic Error (MSLE)

msle = mean_squared_log_error(y_test, y_pred1)
print('Mean Squared Logarithmic Error (MSLE):', msle)
```

```
Mean Squared Error: 3.174097353976104
Mean Absolute Error (MAE): 1.4607567168117601
R-squared (R2) Score: 0.899438024100912
Adjusted R-squared Score: 0.8910578594426547
Mean Squared Logarithmic Error (MSLE): 0.030754526017544713
```

In [18]:
```python
# Create a decision tree regressor

model2 = DecisionTreeRegressor() # Train the decision tree model
model2.fit(X_train, y_train)

# Predict the sales values for the test set

y_pred2 = model2.predict(X_test)

# Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred2)
print('Mean Squared Error:', mse)

# Calculate Mean Absolute Error (MAE)

mae = mean_absolute_error(y_test, y_pred2)
print('Mean Absolute Error (MAE):', mae)

# Calculate R-squared (R2) Score

r2 = r2_score(y_test, y_pred2)
print('R-squared (R2) Score:', r2)

# Calculate Adjusted R-squared Score

n = X_test.shape[0] # Number of samples
p = X_test.shape[1] # Number of predictors
adjusted_r2 = 1 - ((1 - r2) * (n - 1)) / (n - p - 1)
print('Adjusted R-squared Score:', adjusted_r2)

# Calculate Mean Squared Logarithmic Error (MSLE)

msle = mean_squared_log_error(y_test, y_pred2)
print('Mean Squared Logarithmic Error (MSLE):', msle)
```

```
Mean Squared Error: 1.5072500000000004
Mean Absolute Error (MAE): 0.8975000000000002
R-squared (R2) Score: 0.9522471993544778
Adjusted R-squared Score: 0.9482677993006843
Mean Squared Logarithmic Error (MSLE): 0.01441358318167547
```

I used both of these models because they have different strengths. Linear regression assumes a linear relationship between features and the target, provides interpretable coefficients, and is sensitive to outliers. Decision tree regressor makes no assumptions about linearity, can handle non-linear relationships, and may be less interpretable but can handle outliers and capture complex patterns. I chose these two because they are opposites when it comes to linear relatinships. We have 2 extremes. One assuming a linear relationship and another making no assumptions.

As you can see with the results that the decision tree regressor is a better model because of the MSE results. The MSE measures the average squared difference between the predicted and actual values. It means that the predictions are off by the output. The lower the output of MSE the closer the model is to the actual sales values.

The MAE calculates the average absolute difference between the predicted and actual values. With a MAE of 0.955, it means that, on average, the absolute difference between the predicted and actual values of the target variable is 0.955.

The R-squared score measures the proportion of the variance in the target variable that is explained by the model. An R2 score of 0.9498 indicates that approximately 94.98% of the variance in the target variable can be explained by the linear regression model. The higher the r^2 score is the better the model is. The r^2 result for both is about the same.

The adjusted R-squared adjusts the R2 score by taking into account the number of predictors and the sample size. It penalizes the addition of unnecessary predictors that do not improve the model's performance. Higher adjusted R2 scores suggest a stronger fit of the model to the data. The R^2 was higher in the decision tree as well.

The MSLE measures the average logarithmic squared difference between the predicted and actual values. It is useful when the target variable has exponential growth patterns or spans a large range. The MSLE value of 0.0148 indicates that, on average, the logarithmic squared difference between the predicted and actual values is 0.0148.

In [ ]: