Pipelines Project

Insight group: Amal Almutairi, Nouf Aljohani,
Rahaf Alzahrani, Rawan Alsudias & Salha Nasser

Agenda

- Overview of the Dataset
- EDA
- Pipelines
  - Linear Regression Model
  - Logistic Regression Model
- Results

Overview of the Dataset

A simulated data set containing sales of child car seats at 400 different stores.

EDA

Replace Zero

```
df.describe()
```

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.900000
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.620528
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.000000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.000000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.000000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.000000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.000000

Replace Zero

```
df[df['Sales']==0]
```

✓ 0.4s

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
174	0.0	139	24	0	358	185	Medium	79	15	No	No

```
df.iloc[174]['Sales']=df['Sales'].mean()
```

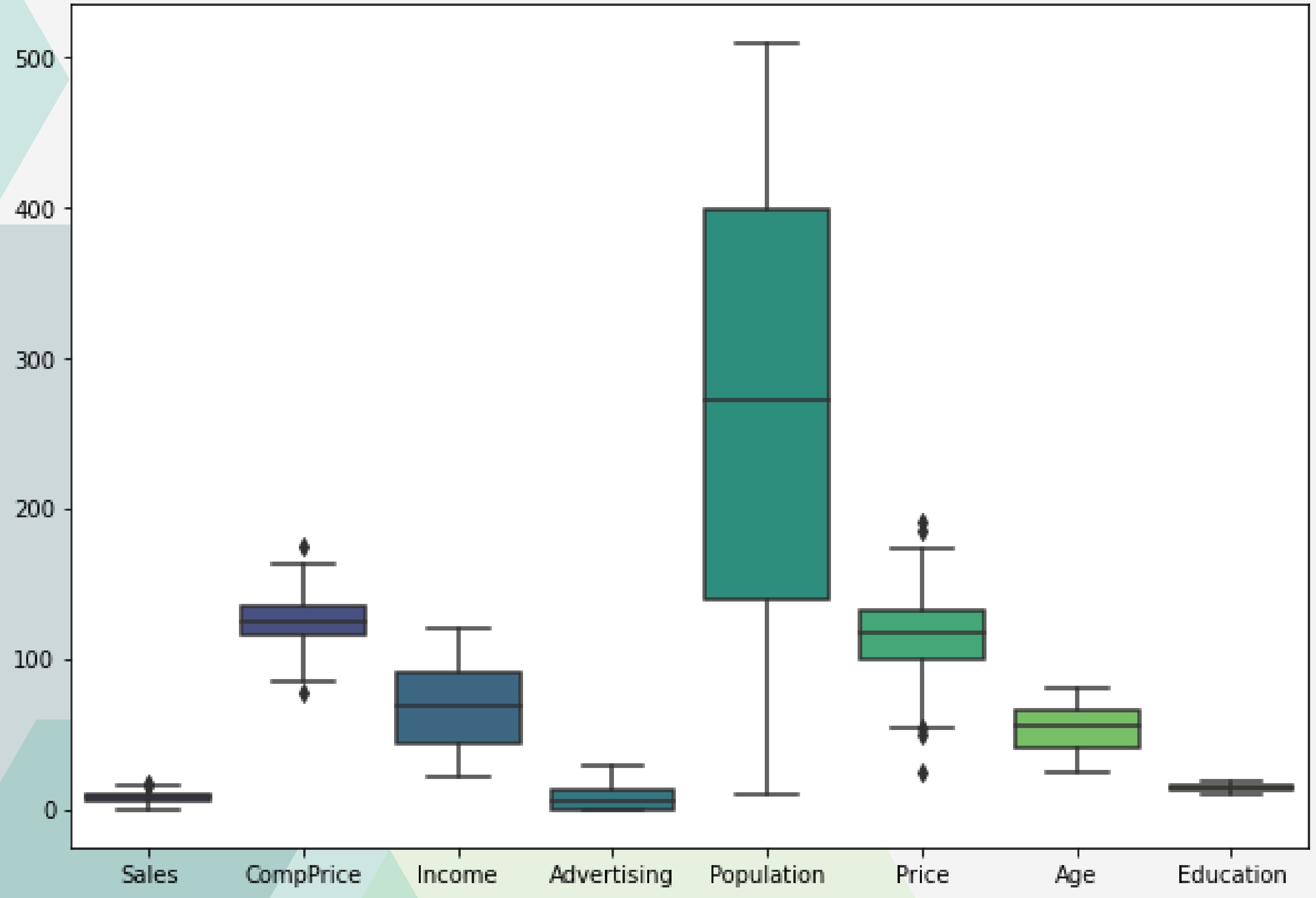
✓ 0.3s

```
df.iloc[174]['Sales']
```

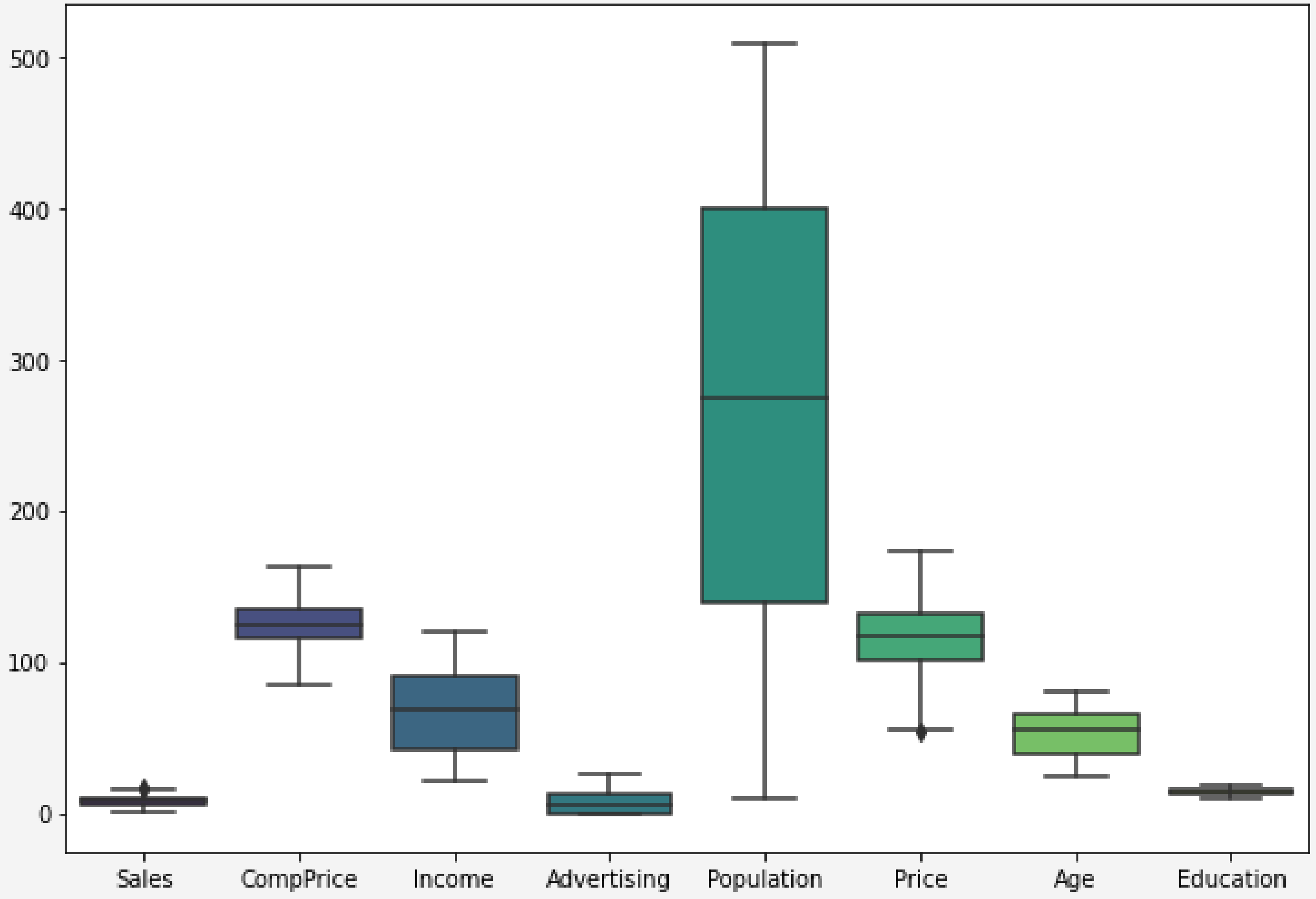
```
10.66
```

Outliers

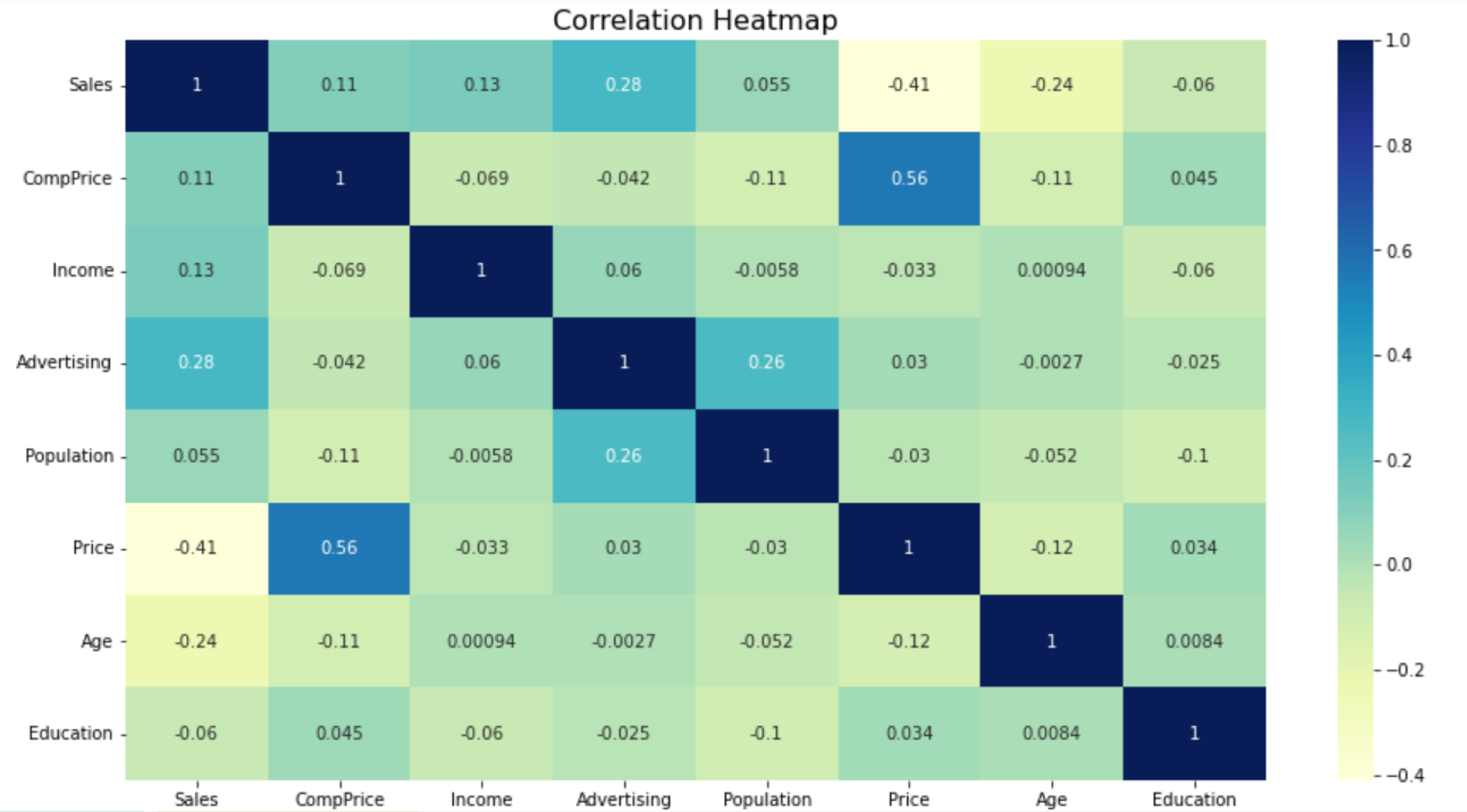
Before



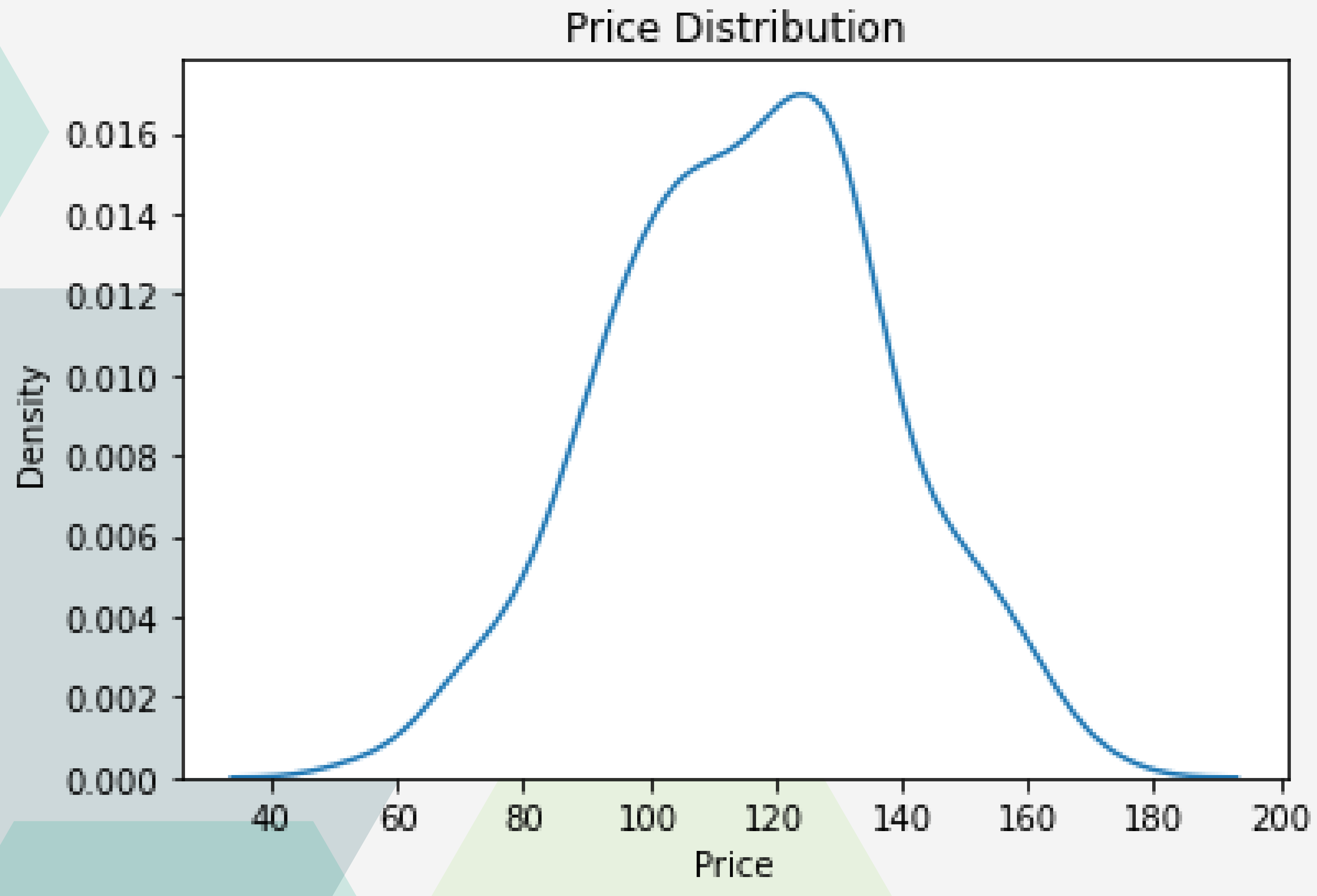
AFTER

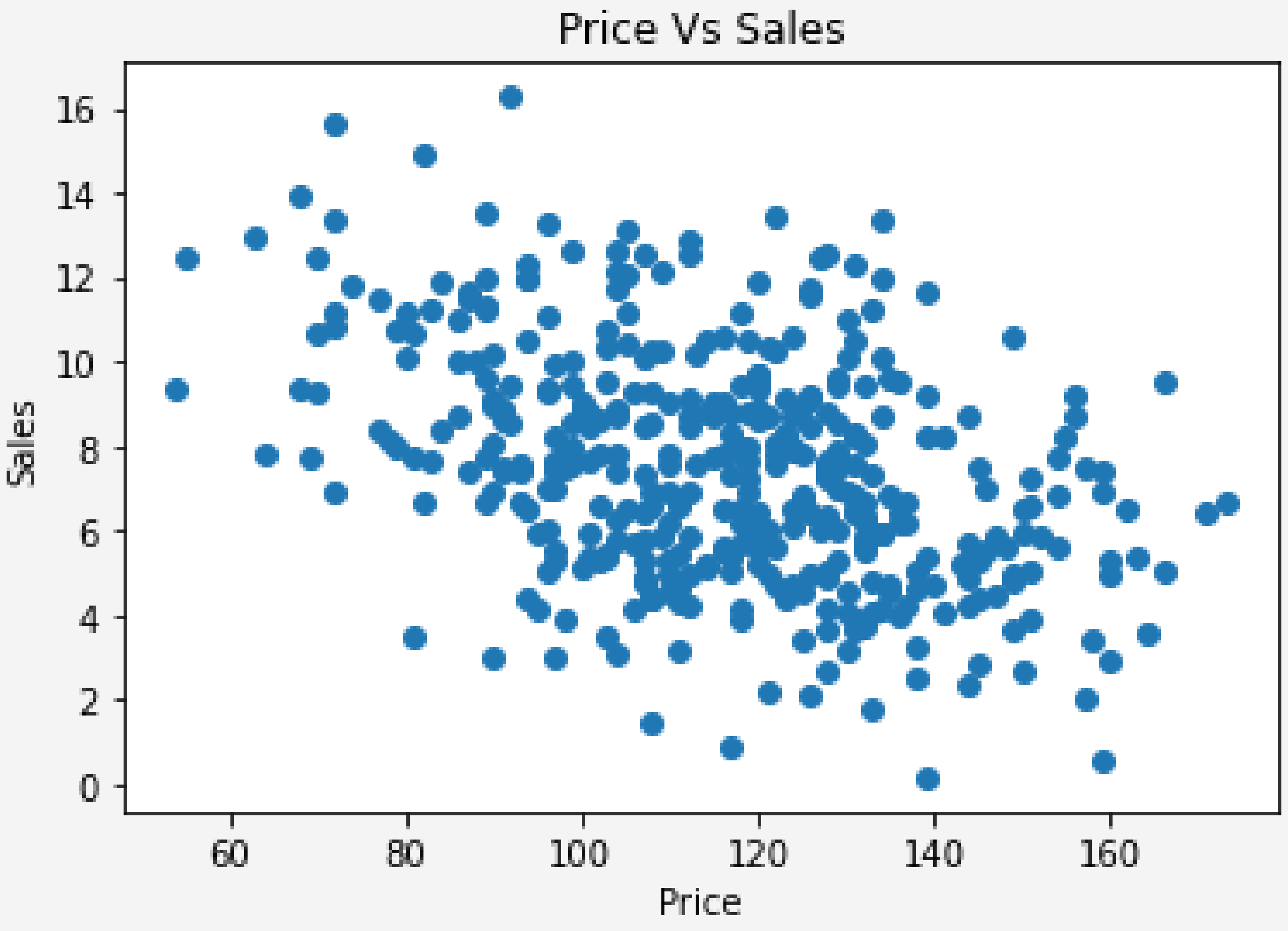


Relationships between variables

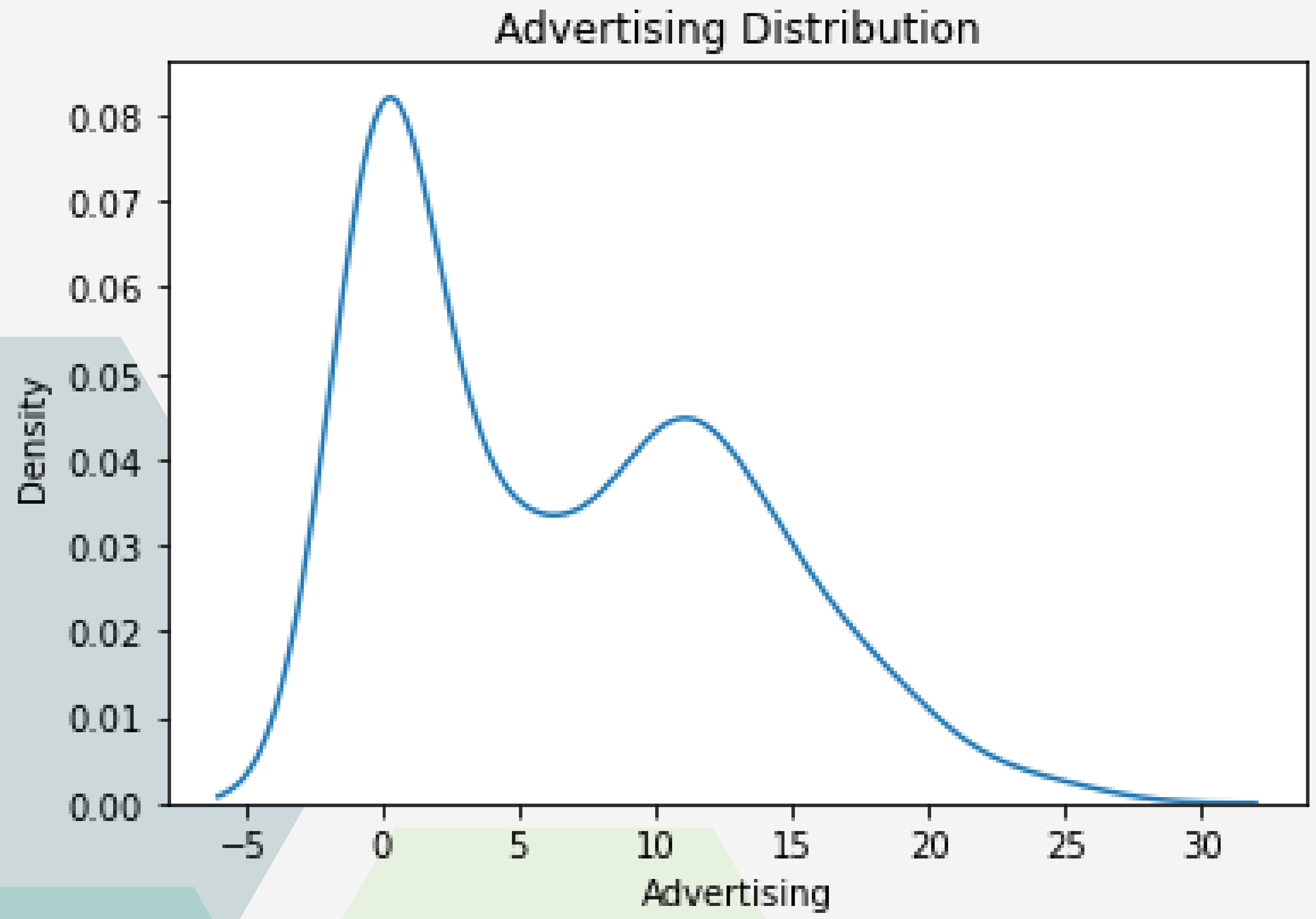


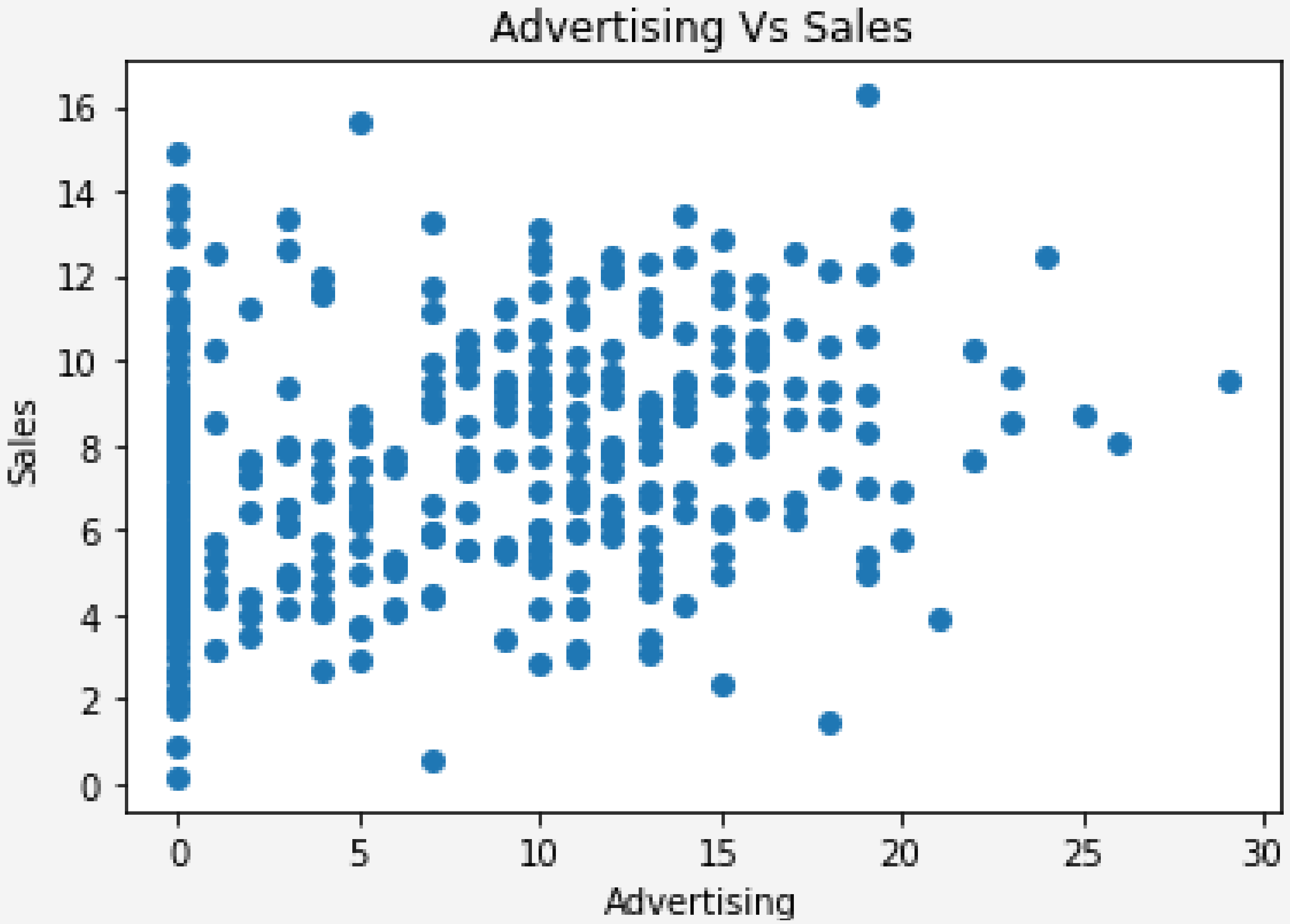
Numeric variable : Price



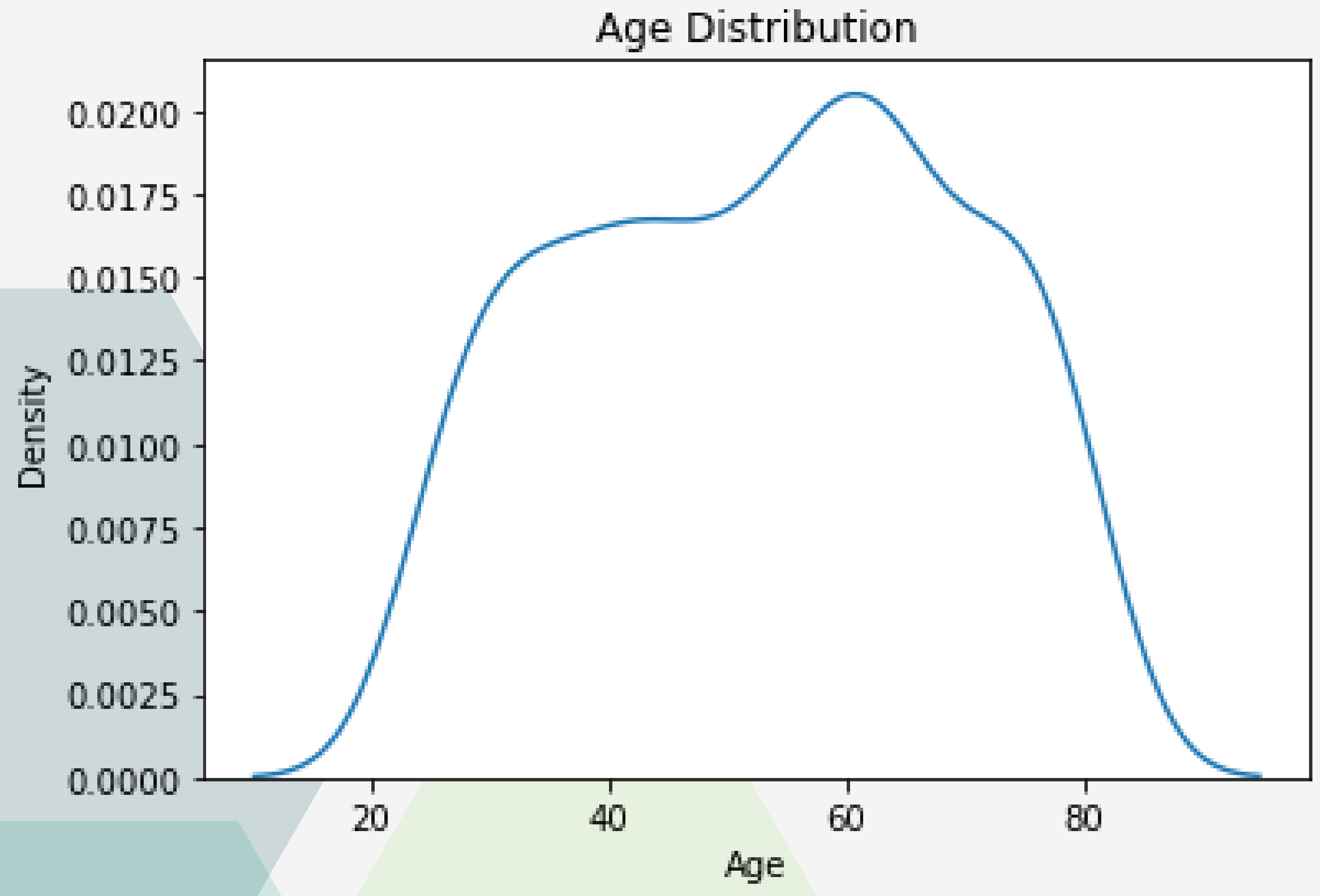


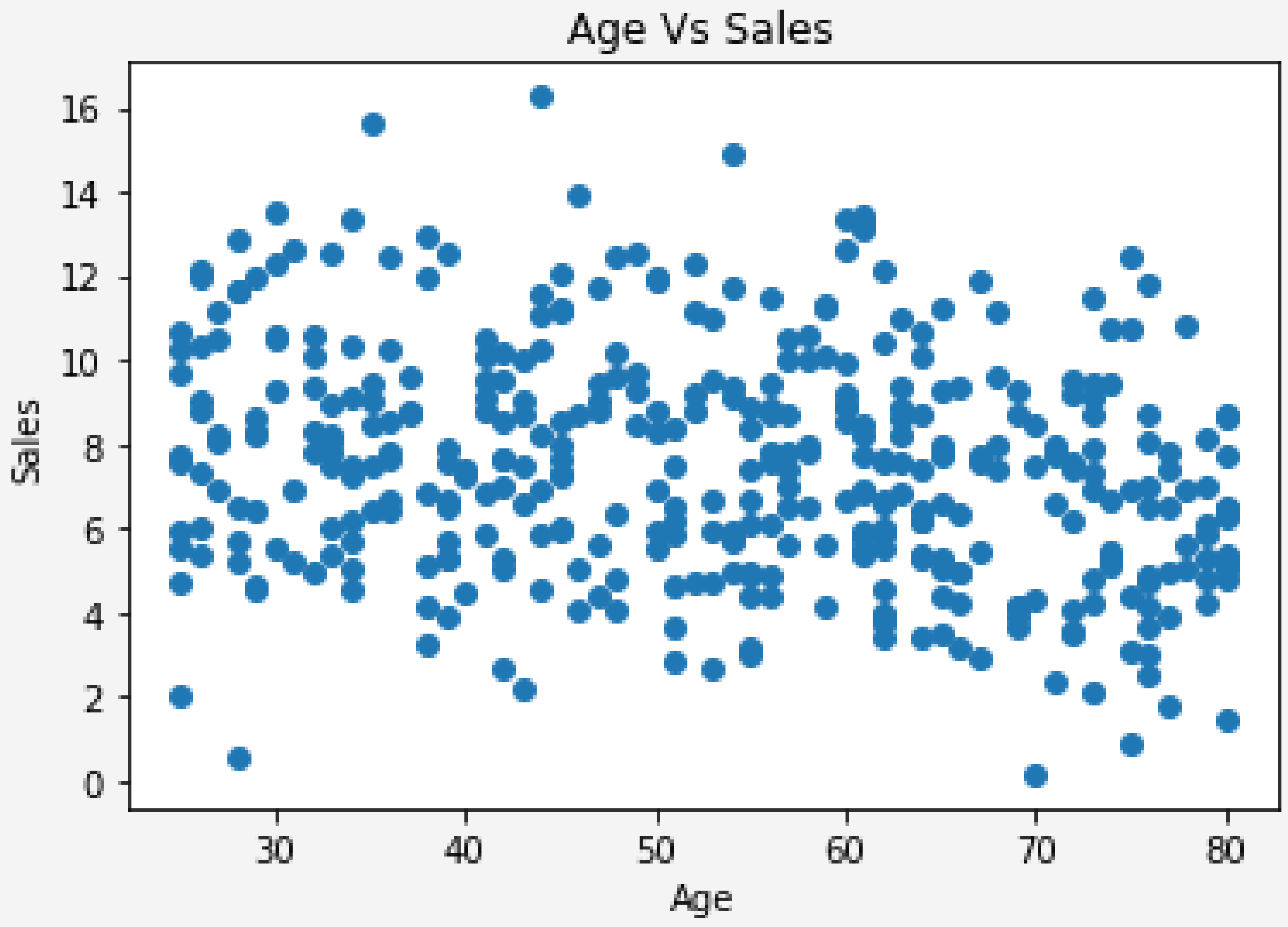
Numeric variable : Advertising



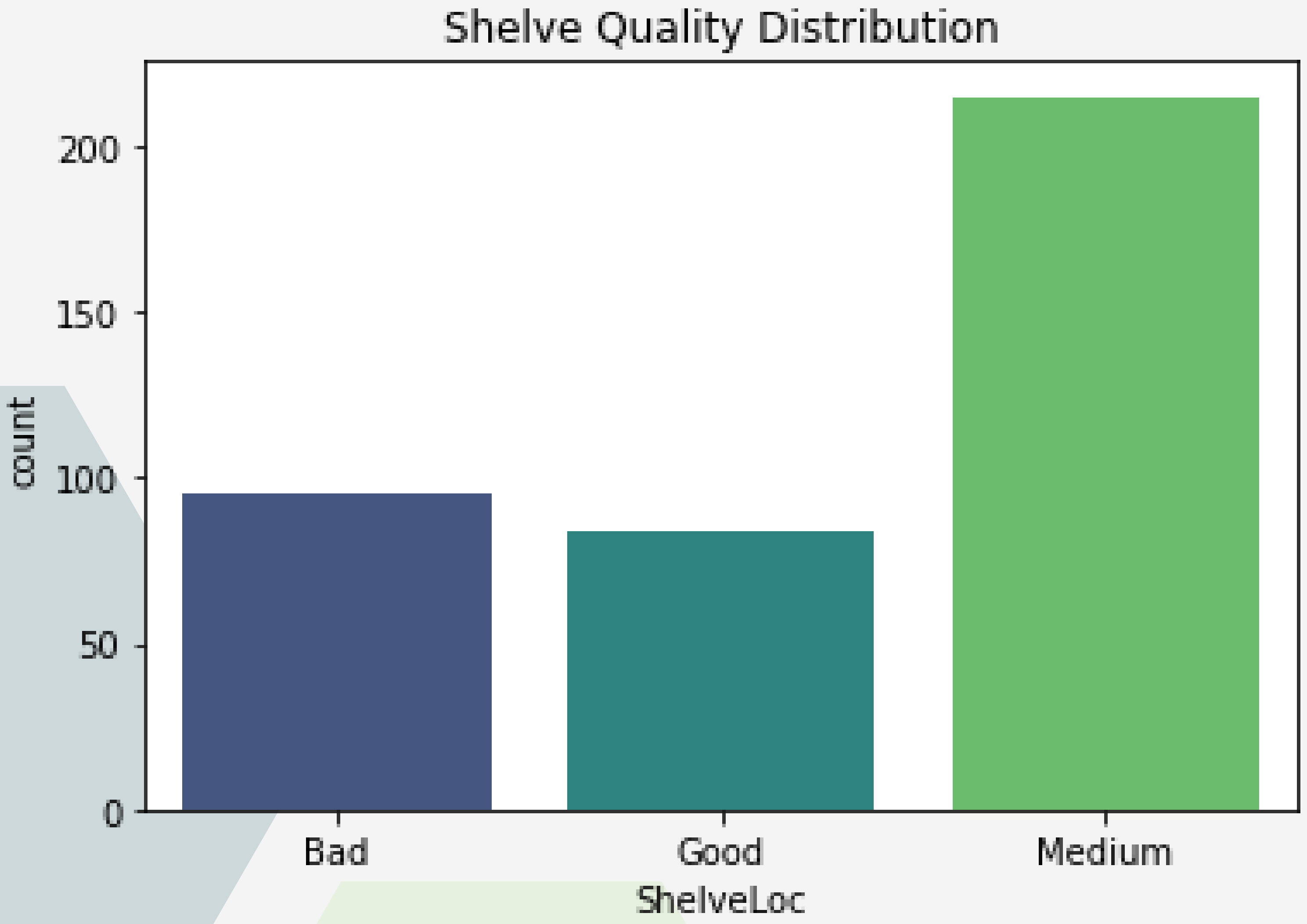


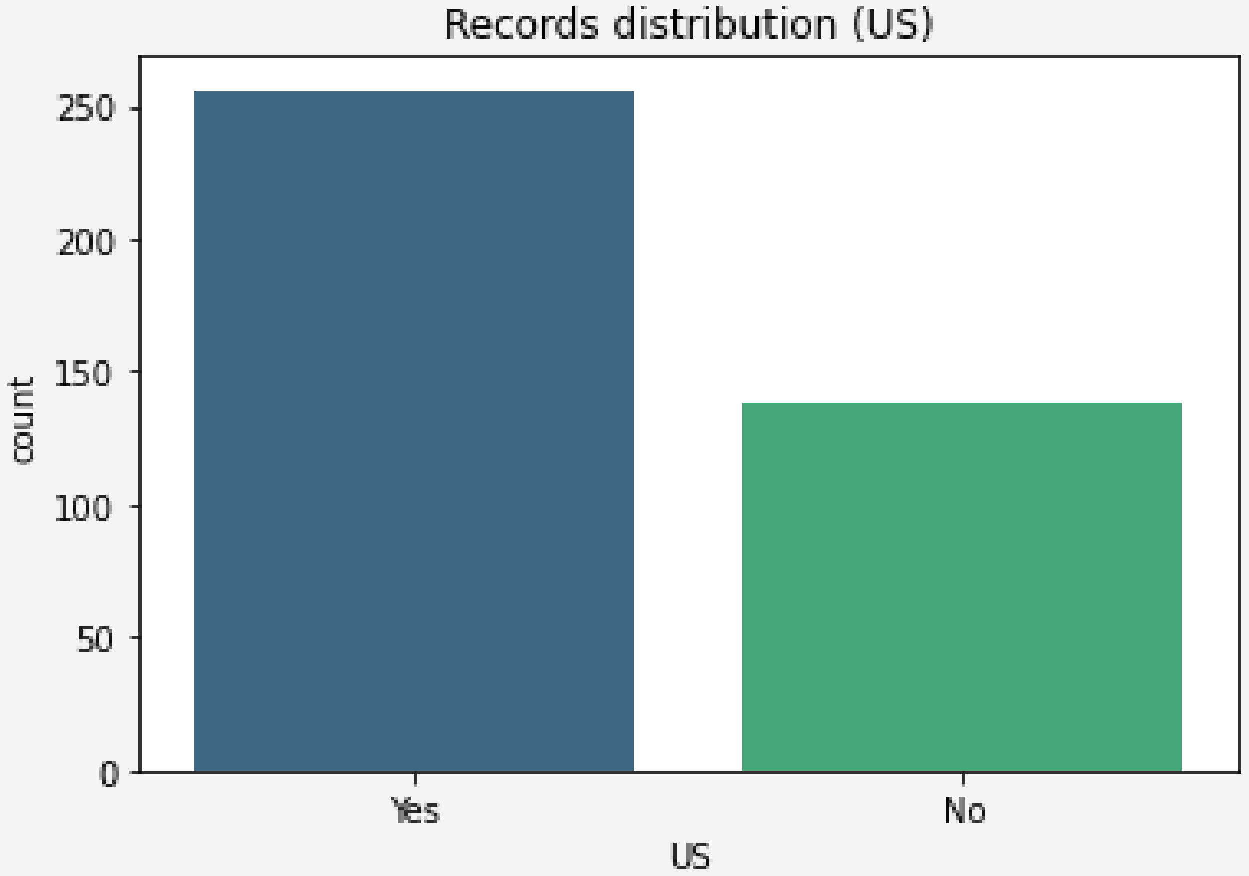
Numeric variable : Age

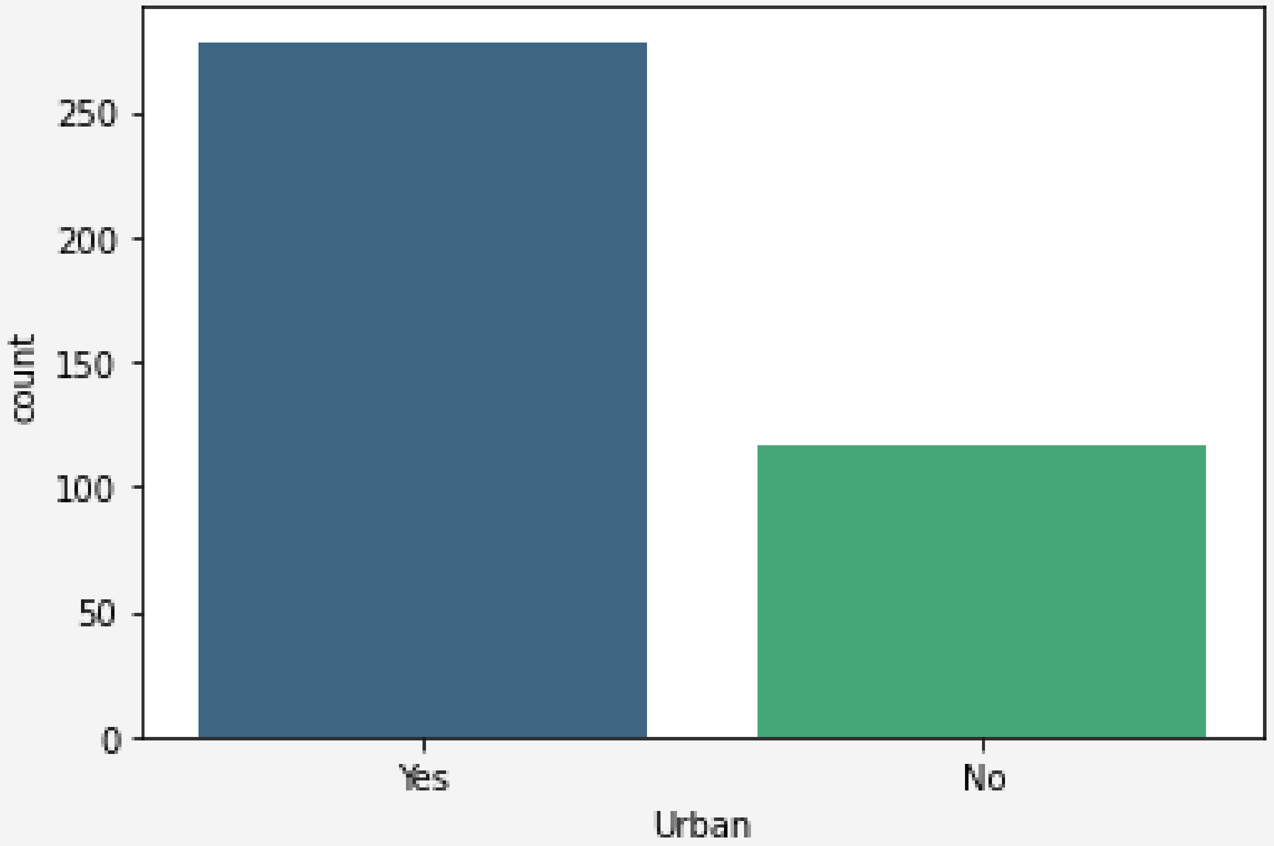




Categorical Variables







Pipelines

Linear Regression Model

```
target = "Sales"
# feature set --> it cannot have the target
X = df.drop(target, axis=1)
# target set
y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.8, random_state=20)
```

```python
# return only numeric columns names
numeric_features = X_train.describe().columns

# Return only categorical names
categorical_features = X_train.describe(exclude="number").columns
```

```python
# Create a transformer for numeric columns
numeric_transformer = Pipeline(
    steps=[
        ('scaler', StandardScaler())
    ]
)

# Create Transformer for categorical data
categorical_transformer = Pipeline(
    steps=[
        # most_frequent --> mode
        ('one_hot', OneHotEncoder(handle_unknown='ignore'))
        # Ignore unseen categorical in transform step not seen in fit_transform
    ]
)

# Create a preprocessor transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

```python
# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf = Pipeline(
    steps=[
        ('preprocessor', preprocessor),
        ('classifier', LinearRegression())
    ]
)

clf.fit(X_train, y_train)

print(f"model score: {clf.score(X_test, y_test)}")
# model score: 0.8597548995008094
```

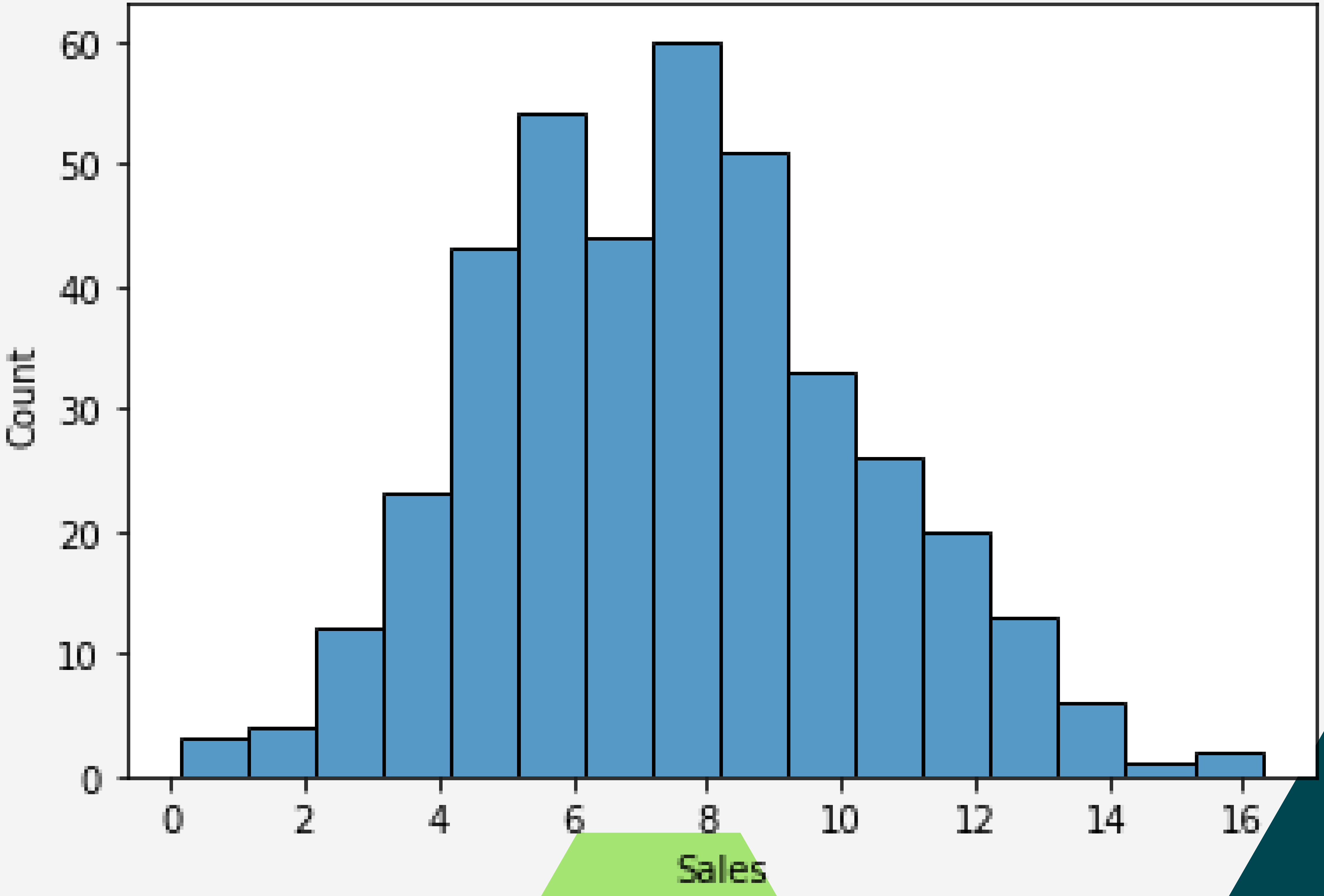
```
predictions = clf.predict(X_test)

error=mean_absolute_error(y_true=y_test, y_pred=predictions)
# 0.8392990661154147
```

```python
# Df for the test points
test_dataset1 = pd.DataFrame(X_test)
test_dataset1.reset_index(inplace=True)
test_dataset1['Actual Sales'] = y_test.to_numpy()

test_dataset1['Predict Sales'] = predictions
percentage = np.zeros(test_dataset1.shape[0])

for i in range(0,test_dataset1.shape[0]):
    percentage[i]= abs(((test_dataset1.loc[i, 'Actual Sales'] - test_dataset1.loc[i, 'Predict Sales'])/
test_dataset1.loc[i, 'Actual Sales'])*100)

test_dataset1['percentage_diff %'] = percentage

# here we're classifying the result to True, or False based on a 25% tolerance.
test_dataset1['Label'] = np.isclose(test_dataset1['Actual Sales'], test_dataset1['Predict Sales'], rtol=0.25)
```

```
test_dataset1['Label'].value_counts()
# True      66
# False     13
```

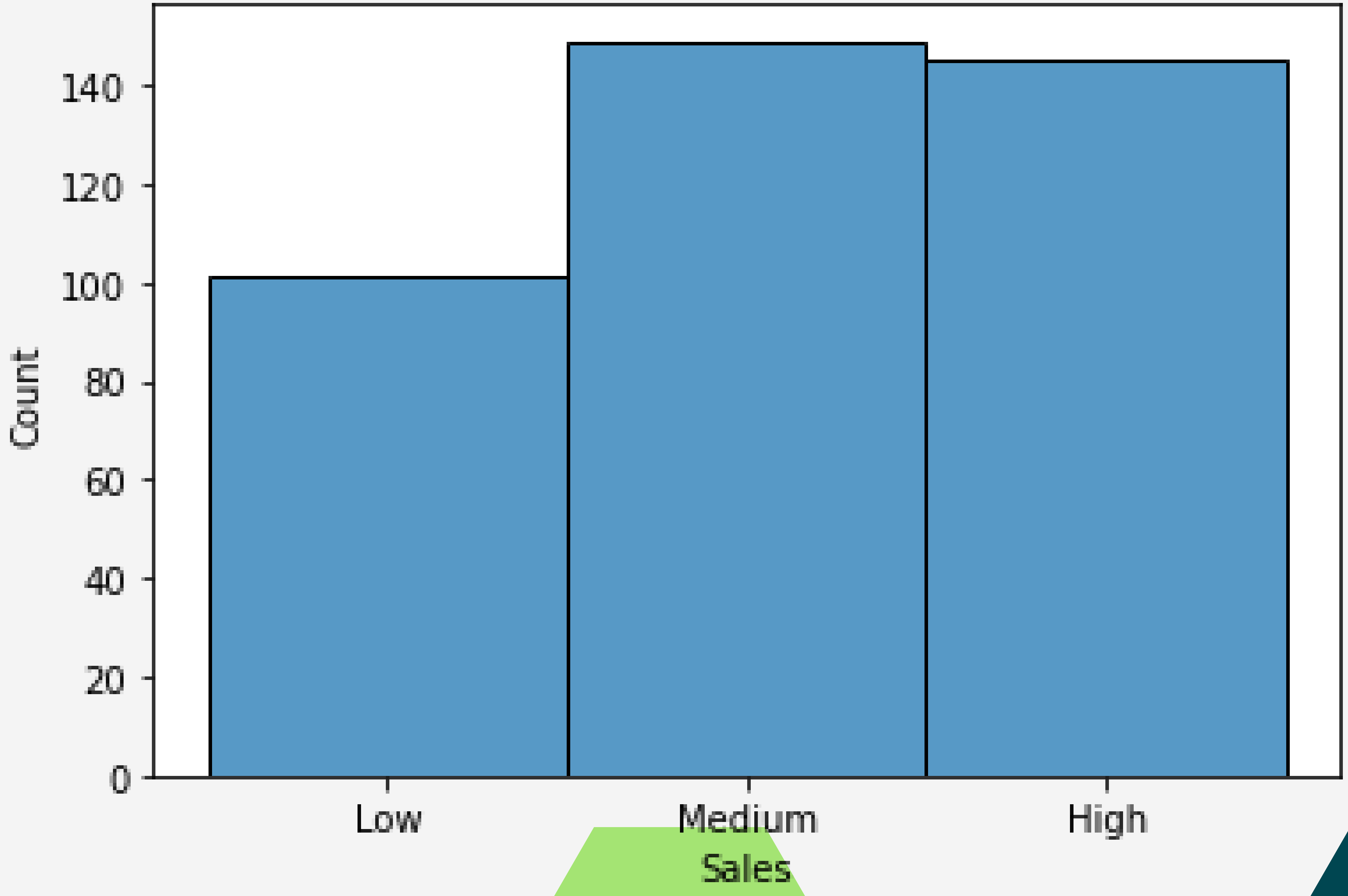
Logistic Regression Model

Histograms of sales



```python
# categorize tip column with values between 0 and 10.
bins = [-1, 5.5, 8.4,16.3]
labels = ['Low', 'Medium', 'High']
df2['Sales'] = pd.cut(df2['Sales'], bins = bins,
labels=labels)
```

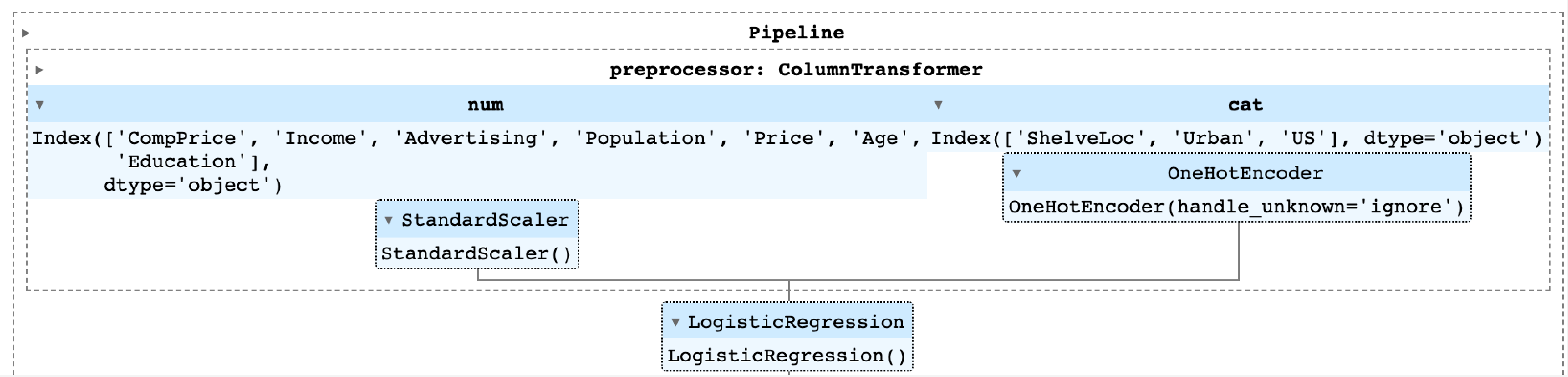
Distribution of Sales Labels



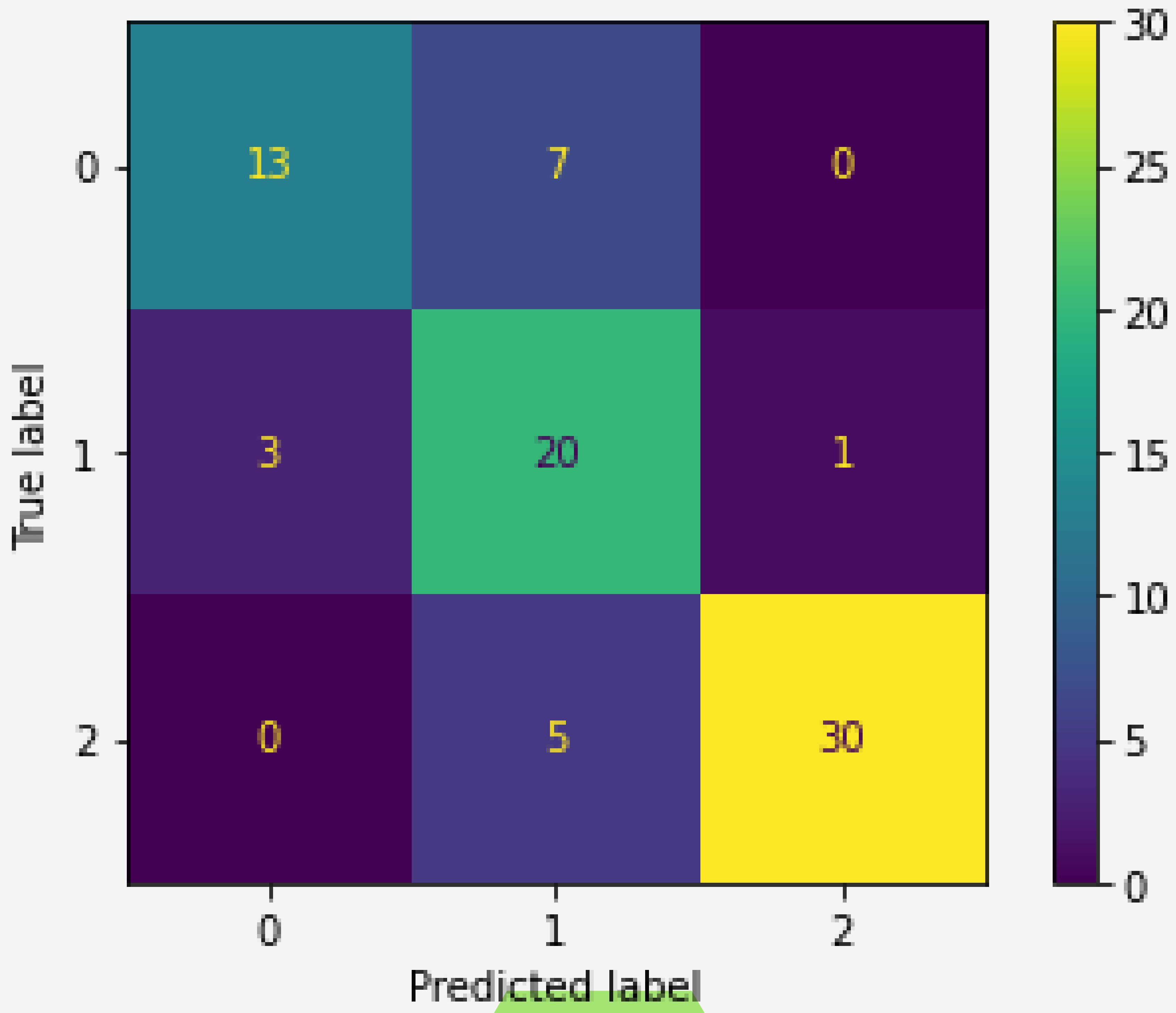
```python
target2 = "Sales Label"
# feature set --> it cannot have the target
X2 = df2.drop(target2, axis=1)
# target set
y2 = df2[target2]

X_train2, X_test2, y_train2, y_test2 = train_test_split(X2,y2,train_size=0.8, random_state=20)
```

```python
#get numric features
numeric_features2 = X_train2.describe().columns

numeric_features2

#output
Index(['CompPrice', 'Income', 'Advertising', 'Population', 'Price', 'Age',
       'Education'],
      dtype='object')


# Return only categorical names

categorical_features2 = X_train2.describe(exclude="number").columns

categorical_features2

#output
Index(['ShelveLoc', 'Urban', 'US'], dtype='object')
```

```python
# Create a transformer for numeric columns

numeric_transformer2 = Pipeline(
    steps=[


        ('scaler', StandardScaler())
    ]
)

# Create Transformer for categorical data

categorical_transformer2 = Pipeline(
    steps=[
        # most_frequent --> mode

        ('one_hot', OneHotEncoder(handle_unknown='ignore')) # Ignore unseen categorical in transform
step not seen in fit_transform
    ]
)

# Create a preprocessor transformer
preprocessor2 = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer2, numeric_features2),
        ('cat', categorical_transformer2, categorical_features2)
    ]
)

# Append classifier to preprocessing pipeline.
# Now we have a full prediction pipeline.
clf2 = Pipeline(
    steps=[
        ('preprocessor', preprocessor2),
        ('classifier', LogisticRegression())
    ]
)
```



Accuracy :0.7974


13
7
0
3
20
1
0
5
30
True label
0
1
2
Predicted label
0
1
2
30
25
20
15
10
5
0

Results

```python
lgmodel= (test_dataset['Actual Label'] == test_dataset['Predict Label']).value_counts() # MODEL 1
lrmodel= test_dataset1['Label'].value_counts() # MODEL 2



print('Results Overview:\n')

print('With the Same Test Data Points, for Both Models:')
print(f'Linear Regression Model :\n True: {lrmodel[1]}\n False: {lrmodel[0]}')
print(f'Logistic Regression Model :\n True: {lgmodel[1]}\n False: {lgmodel[0]}')
print('Accuracy Based on these Test Points is:')
print(f'Linear Regression Model Accuracy: {round(lrmodel[1]/80*100,2)}%')
print(f'Logistic Regression Model Accuracy: {round(lgmodel[1]/80*100,2)}%')

print('-----------------------------------------------------')


print('Linear Regression Model  Performed Better than Logistic Regression Model!')
```

Linear Regression Model :

True: 66

False: 13

Logistic Regression Model :

True: 63

False: 16

Accuracy Based on these Test Points is:

Linear Regression Model Accuracy: 82.5%

Logistic Regression Model Accuracy: 78.75%

Reference:

https://www.kaggle.com/code/akashchola/decision-tree-for-classification-regression/data


Thank you!