Data learning: Multiple Linear Model vs Regression Tree

Example 1: Single Outcome Variable and Two Covariates

Data source: Simulation by R or Python programming language

Data:

#	Ads Budget(X)	Sales(Y)	Ads_Budget_Interaction
1	10	15	15
2	20	25	50
3	30	30	90
4	40	35	140
5	50	40	200

Step 1: Formulate your research question based on this data set.

Is Advertisement budget and Ads Budget Interaction an important factor influencing sales?

Step 2: Define data role and data type.

Variable name	Data role	Data type
Advertisement budget	X	Continuous
Sales	Y	Continuous
Ads Budget Interaction	X	Continuous

Step 3: Select an appropriate method to analyze the data e.g., statistical learning, machine learning.

Multiple linear regression is used to analyze the data because there are two continuous predictor and one continuous outcome.

Regression tree is used to analyze the data because we trying to group the data.

Step 4: Collect data.

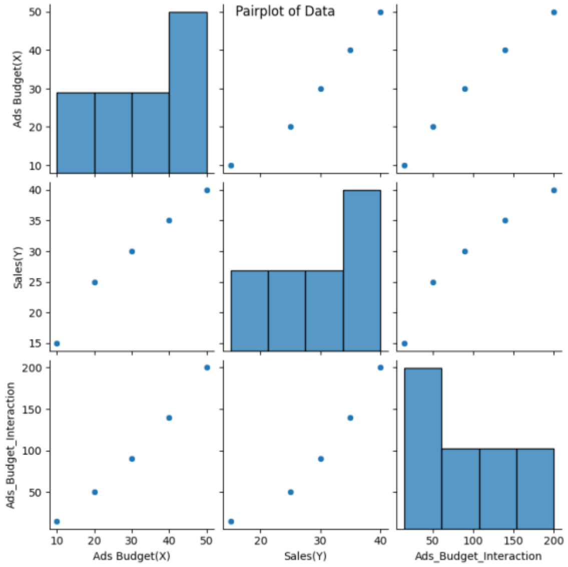
A simulated data set from R or Python programming language is used for this example.

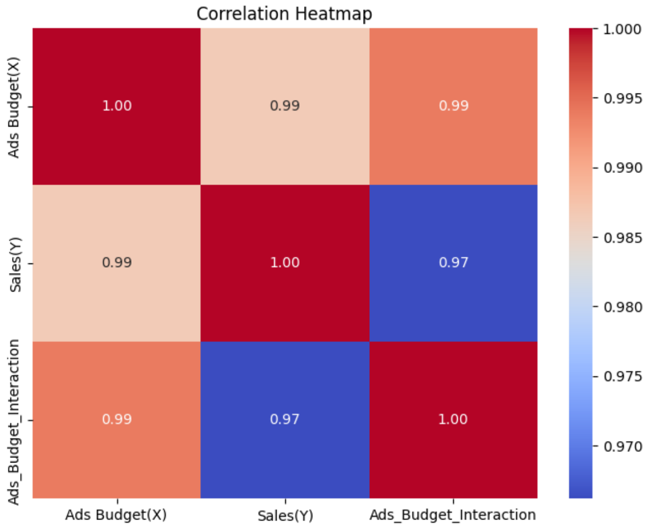
Step 5: Explore your data using numerical summary and graphs

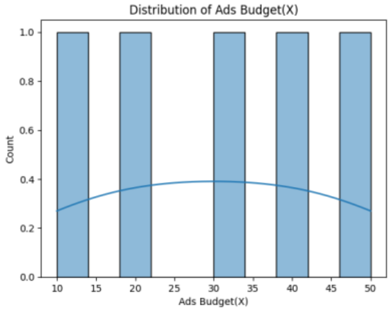
Descriptive statistics:

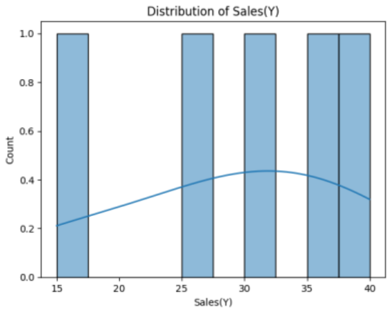
Variable name	N	Min	Max	Mean	Median	Sd	variance
Advertisement budget	5	10	50	30	30	15.811388	250
Sales	5	29	40	29	30	9.617692	92.5
Ads Budget Interaction	5	99	200	99	90	73.177866	5355

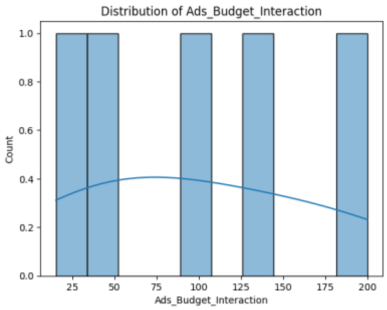
Show your graphs:

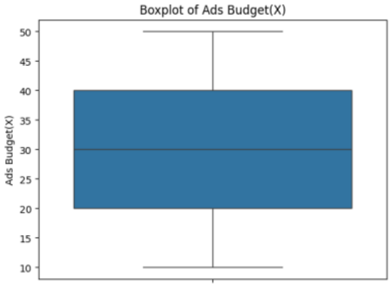


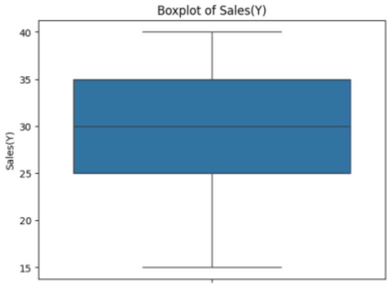


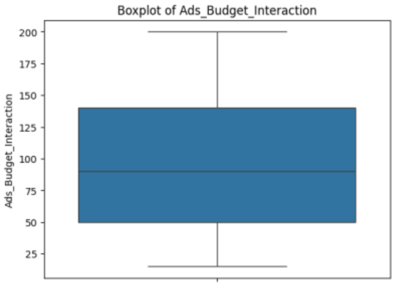












Interpret:

1. Histograms
   - Ads Budget (X): The budget values are evenly distributed across the range, showing minimal skewness.
   - Sales (Y): Sales values are slightly concentrated toward higher amounts, with fewer instances at the lower end.

   - Ads Budget Interaction: Interaction terms increase proportionally with Ads Budget (X), indicating a strong dependency.

2. Scatter Plot (Ads Budget vs. Sales)
   - There is a noticeable positive linear association between Ads Budget (X) and Sales (Y).
   - As the advertisement budget increases, sales generally rise, suggesting Ads Budget (X) is a strong predictor of Sales (Y).
3. Box Plots
   - Ads Budget (X): No major outliers, with values distributed relatively evenly.
   - Sales (Y): The data is slightly skewed toward higher values, without significant outliers.
   - Ads Budget Interaction: The interaction term consistently increases, aligning with the upward trend observed in Ads Budget (X).
4. Heatmap (Correlation Matrix)
   - Ads Budget (X) and Sales (Y):
   - They exhibit a strong positive correlation, supporting the relationship highlighted in the scatter plot.
   - Ads Budget Interaction: This variable shows an even stronger correlation with both Ads Budget (X) and Sales (Y), as it is derived from the budget and reflects its influence.

Multiple Linear model

Step 6: Fit the model.

F-test

- Null Hypothesis: All regression coefficients are equal to zero.
- Alternative Hypothesis: At least one regression coefficient is not zero.
- The F-statistic of 95.2 indicates a very strong relationship between the independent variables and the dependent variable. This result suggests that the overall model is robust. However, individual predictors should be evaluated for significance using their respective p-values and by validating model assumptions.

t-test

- Null Hypothesis: The predictor variable has no impact on the dependent variable.
- Alternative Hypothesis: The predictor variable affects the dependent variable.

1. Constant (Intercept): t = 1.368. The intercept does not appear to be statistically significant since the t-value is small, and its p-value is likely greater than 0.05.
2. Ads Budget (X): t = 3.285. This relatively large t-value suggests that Ads Budget (X) has a significant positive effect on Sales (Y), with its p-value likely below 0.05. As the budget increases, sales are expected to rise substantially.
3. Ads Budget Interaction: t = -1.789. The negative t-value suggests a weak or marginally significant negative effect on Sales (Y), with its p-value around 0.05.

Step 7: Check standard assumptions.

1. Independence of Errors

- Using the Durbin-Watson (DW) test, DW = 2.731 suggests negative autocorrelation in the residuals. Negative autocorrelation can impact the efficiency of the regression coefficients and lead to misleading results.

2. Multicollinearity

- If VIF > 10, there is severe multicollinearity, which is a significant issue.

Step 8: Evaluate model accuracy.

- R^2: 98.48% of the variability in Sales (Y) is explained by the model, indicating a very strong fit.
- AIC (18.88): AIC balances the goodness of fit and model complexity. A lower value indicates a better model.
- BIC (17.71): BIC also evaluates the trade-off between model fit and complexity but penalizes model complexity more heavily than AIC, especially for large sample sizes.

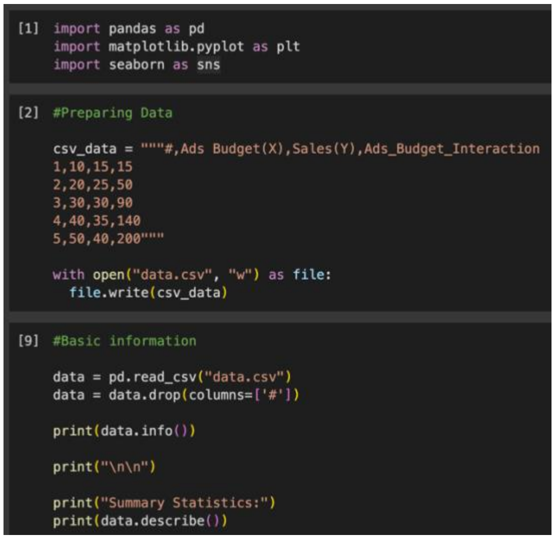
Step 9: Interpret the results.

The multiple regression model demonstrates that:

- Ads Budget (X) significantly increases sales, explaining 98.48% of the variability in Sales (Y).
- The Ads Budget Interaction term shows a weak negative effect on Sales (Y).
- The intercept is not statistically significant.

****

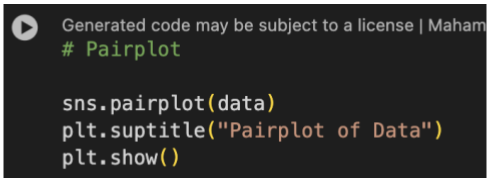
Show your R or Python programming language for binary logistic regression.



```
[1] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
```

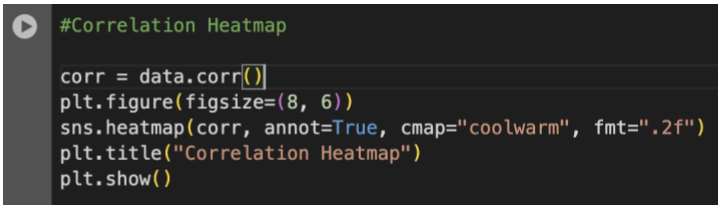
```
[2] #Preparing Data

    csv_data = """#,Ads Budget(X),Sales(Y),Ads_Budget_Interaction
    1,10,15,15
    2,20,25,50
    3,30,30,90
    4,40,35,140
    5,50,40,200"""

    with open("data.csv", "w") as file:
      file.write(csv_data)
```

```
[9] #Basic information

    data = pd.read_csv("data.csv")
    data = data.drop(columns=['#'])

    print(data.info())

    print("\n\n")

    print("Summary Statistics:")
    print(data.describe())
```

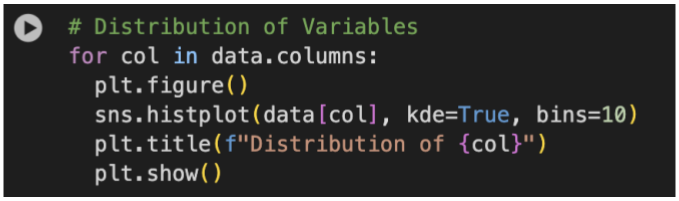


Generated code may be subject to a license | Maham

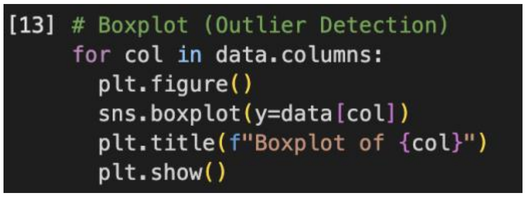
```
# Pairplot

sns.pairplot(data)
plt.suptitle("Pairplot of Data")
plt.show()
```



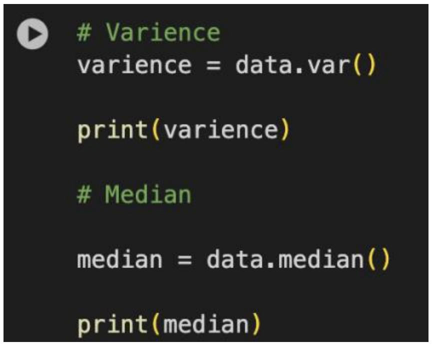
```
#Correlation Heatmap

corr = data.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



```
# Distribution of Variables
for col in data.columns:
  plt.figure()
  sns.histplot(data[col], kde=True, bins=10)
  plt.title(f"Distribution of {col}")
  plt.show()
```

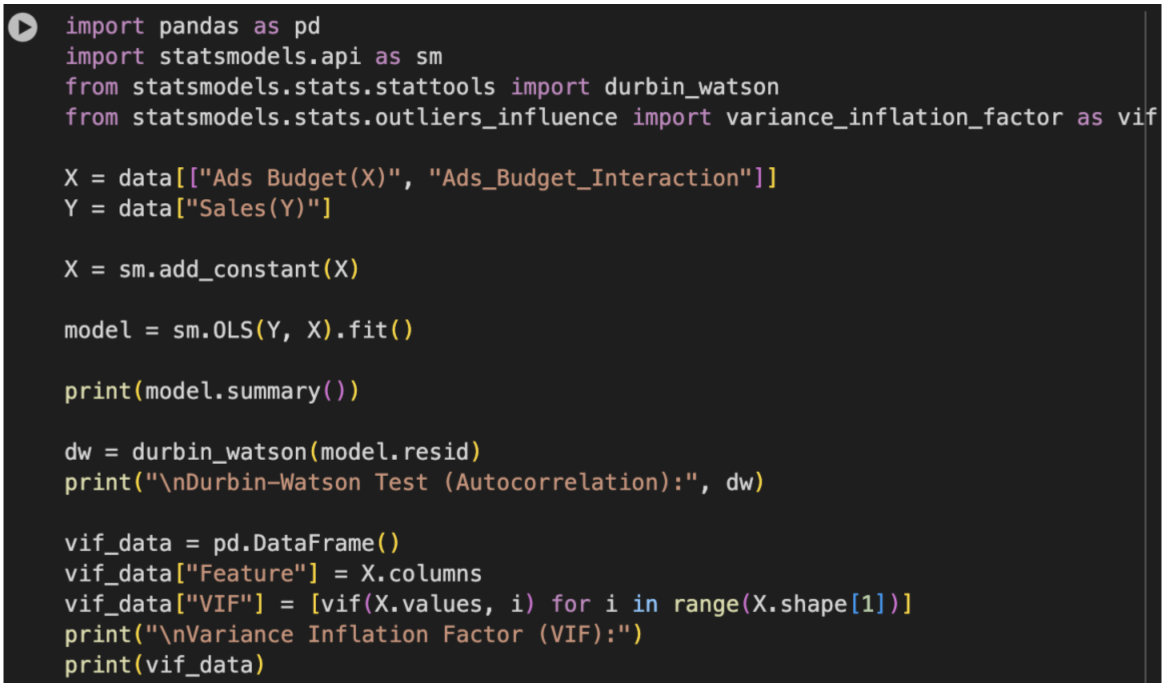


```
[13] # Boxplot (Outlier Detection)
     for col in data.columns:
       plt.figure()
       sns.boxplot(y=data[col])
       plt.title(f"Boxplot of {col}")
       plt.show()
```



```
# Varience
varience = data.var()

print(varience)

# Median

median = data.median()

print(median)
```

Sirada Boonmarn 6609655353

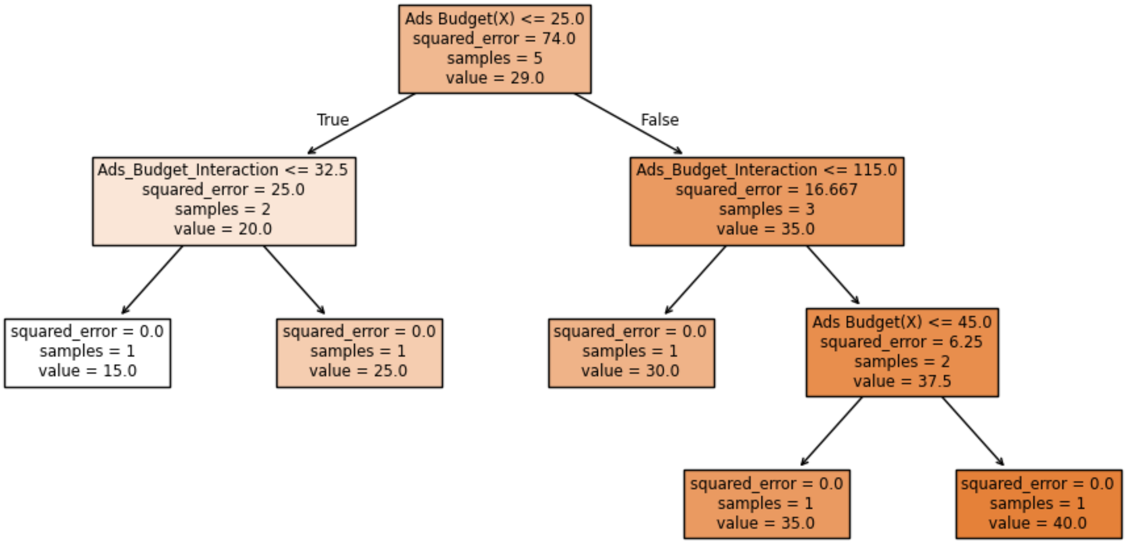


```
import pandas as pd
import statsmodels.api as sm
from statsmodels.stats.stattools import durbin_watson
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif

X = data[["Ads Budget(X)", "Ads_Budget_Interaction"]]
Y = data["Sales(Y)"]

X = sm.add_constant(X)

model = sm.OLS(Y, X).fit()

print(model.summary())

dw = durbin_watson(model.resid)
print("\nDurbin-Watson Test (Autocorrelation):", dw)

vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [vif(X.values, i) for i in range(X.shape[1])]
print("\nVariance Inflation Factor (VIF):")
print(vif_data)
```

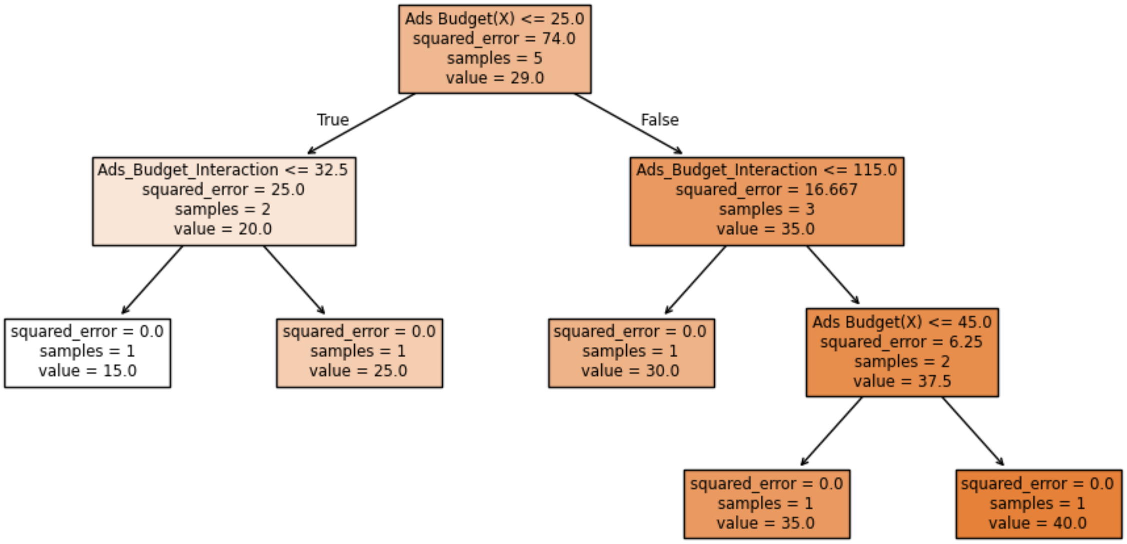
Regression tree

Step 6: Fit the model.

Regression Tree Analysis (No Pruning):



Regression Tree Analysis (With Pruning):



Step 7: Interpret the results.

The pruned tree is recommended for its balance between simplicity, interpretability, and generalization.

- By limiting the depth, it retains the key splits while avoiding overfitting.
- Although the unpruned tree is highly accurate on the training dataset, its performance on unseen data may degrade due to overfitting.

For practical use cases, the pruned tree provides a better balance of accuracy and reliability, making it the preferred choice for predictions.

Model selection/comparison: Multiple Linear Regression VS Regression Tree

Comparison of MSE and RMSE

1. Multiple Linear Regression (MLR):
    - Mean Squared Error (MSE): 0.7692
    - Root Mean Squared Error (RMSE): 0.8771
2. Pruned Regression Tree (RT):
    - Mean Squared Error (MSE): 0.0000
    - Root Mean Squared Error (RMSE): 0.0000

The pruned regression tree has zero MSE and RMSE, which indicates that it overfits the training data. This is likely due to:

1. The small size of the dataset.
2. Overfitting even after pruning.
3. Lack of variation in target values at the nodes.

Which model would you select for classification?

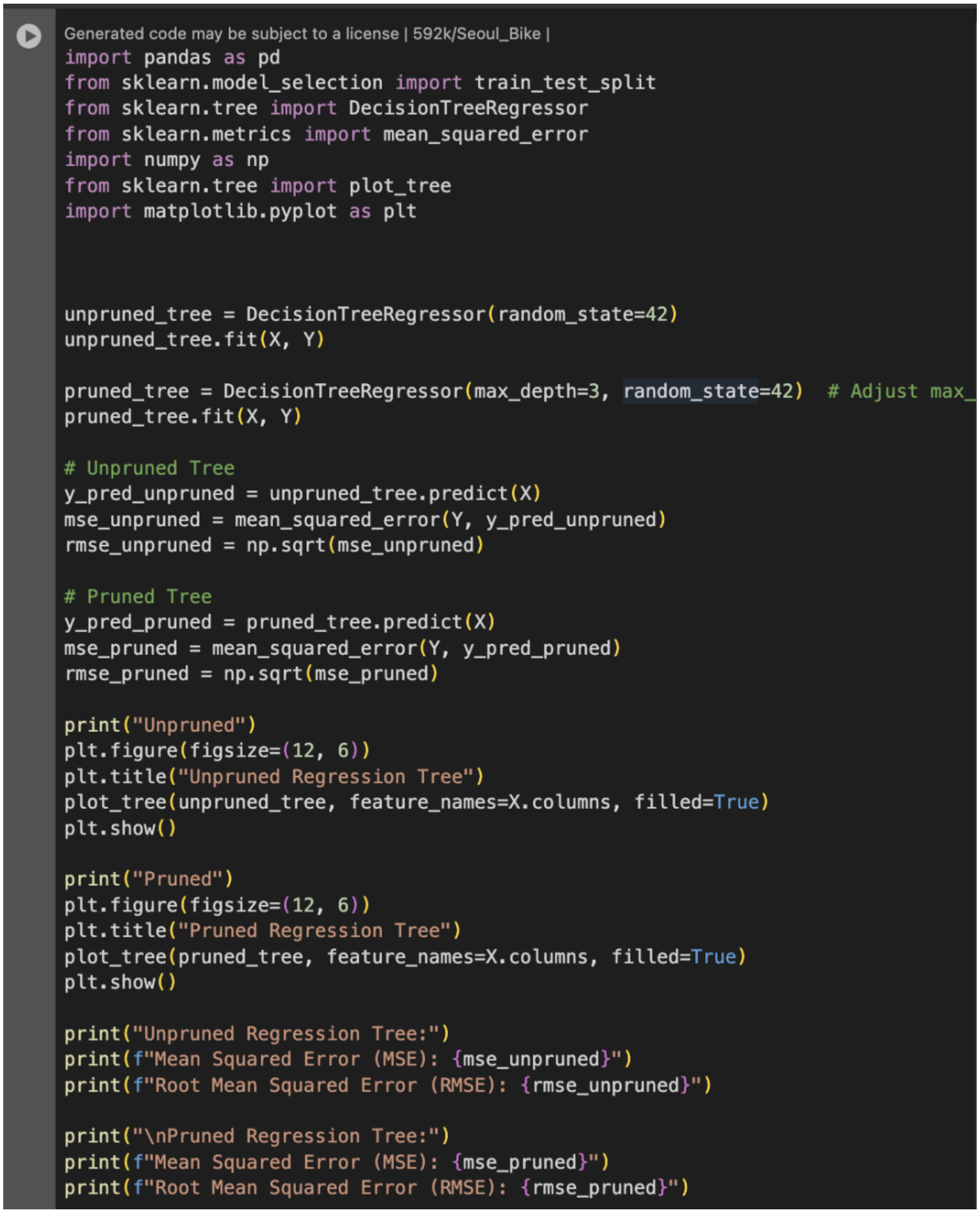
We prefer Multiple Linear Regression (MLR) because:

1. Strong Fit: MLR explains a high percentage of the variability in the target variable.
2. Resilience to Overfitting: Compared to regression trees, MLR is less prone to overfitting, making it more reliable on unseen data.
3. Interpretability: MLR provides clear coefficients that indicate the effect of each predictor.

However, MLR faces challenges with multicollinearity among predictors (indicated by high VIF). In such cases, alternative techniques like Principal Component Regression (PCR) or Factor Analysis (FA) may be more suitable.

****

Show your R or Python programming language for classification tree.



Generated code may be subject to a license | 592k/Seoul_Bike |

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt



unpruned_tree = DecisionTreeRegressor(random_state=42)
unpruned_tree.fit(X, Y)

pruned_tree = DecisionTreeRegressor(max_depth=3, random_state=42)  # Adjust max_
pruned_tree.fit(X, Y)

# Unpruned Tree
y_pred_unpruned = unpruned_tree.predict(X)
mse_unpruned = mean_squared_error(Y, y_pred_unpruned)
rmse_unpruned = np.sqrt(mse_unpruned)

# Pruned Tree
y_pred_pruned = pruned_tree.predict(X)
mse_pruned = mean_squared_error(Y, y_pred_pruned)
rmse_pruned = np.sqrt(mse_pruned)

print("Unpruned")
plt.figure(figsize=(12, 6))
plt.title("Unpruned Regression Tree")
plot_tree(unpruned_tree, feature_names=X.columns, filled=True)
plt.show()

print("Pruned")
plt.figure(figsize=(12, 6))
plt.title("Pruned Regression Tree")
plot_tree(pruned_tree, feature_names=X.columns, filled=True)
plt.show()

print("Unpruned Regression Tree:")
print(f"Mean Squared Error (MSE): {mse_unpruned}")
print(f"Root Mean Squared Error (RMSE): {rmse_unpruned}")

print("\nPruned Regression Tree:")
print(f"Mean Squared Error (MSE): {mse_pruned}")
print(f"Root Mean Squared Error (RMSE): {rmse_pruned}")
```