NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

ARTIFICIAL INTELLIGENCE LAB

NAME	Ayesha Imran
Class	CS-A
Oel	08
Course	Artificial Intelligence
Date	9-December-25
Submitted To	Lec. Ijlal Haider

IN LAB TASKS

TASK 01: For the given dataset "House Price Prediction Dataset.csv", which contains house-related features influencing property prices, complete the following tasks:

- Data Exploration and Preprocessing:
- Perform exploratory data analysis (EDA) to:
- Understand the structure of the dataset (e.g., data types, missing values, and
- summary statistics).
- Visualize relationships between the features (Area, Bedrooms, Bathrooms,
- etc.) and the target variable (Price) using scatter plots, heatmaps, or box plots.
- Preprocess the data by:
- Handling missing values, if any.
- Encoding categorical variables like Condition and Location.
- Normalizing or scaling numerical features like Area and Year Built to ensure
- comparability across models.
- Regression Model Implementation:
- Develop models to predict house prices using the following regression techniques:
- Simple Linear Regression (e.g., predicting Price using Area alone).
- Multiple Linear Regression (using all numerical features).
- Ridge Regression
- Lasso Regression
- Decision Tree Regression
- Random Forest Regression
- Feature Engineering:
- Create new features to capture interactions or non-linear relationships, such as:
- Interaction terms like Area x Condition or Bedrooms x Bathrooms.
- Polynomial features (e.g., $Area^2$) for non-linear regression.
- Performance Evaluation:
- Evaluate the performance of each model using:
- Mean Absolute Error (MAE)

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Coefficient of Determination (R 2)
- Comparison and Insights:
- Compare the models' performance metrics to determine:
- Which features have the strongest influence on Price predictions.
- Which model provides the most accurate and reliable results.
- Use visualization tools to show Actual vs. Predicted Price for each model.

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.feature_selection import SelectKBest, f_regression
import warnings
warnings.filterwarnings('ignore')

# =====================================
# 1. Load and Explore the Data
# =====================================
df = pd.read_csv('House Pridiction Dataset.csv')  # Change path if needed

print("Dataset Shape:", df.shape)
print("\nData Types:\n", df.dtypes)
print("\nMissing Values:\n", df.isnull().sum())
print("\nSummary Statistics:\n", df.describe())

# Target and features
target = 'price'
numerical_features = ['area', 'bedrooms', 'bathrooms', 'stories', 'parking']
categorical_features = ['mainroad', 'guestroom', 'basement', 'hotwaterheating',
                        'airconditioning', 'prefarea', 'furnishingstatus']
```

```python
# =====================================
# 2. Visualizations
# =====================================
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

# Scatter plots
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='area', y='price', hue='furnishingstatus')
plt.title('Price vs Area colored by Furnishing Status')
plt.show()

# Box plots
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='furnishingstatus', y='price')
plt.title('Price Distribution by Furnishing Status')
plt.show()

# =====================================
# 3. Feature Engineering
# =====================================
# Interaction terms
df['area_bedrooms'] = df['area'] * df['bedrooms']
df['bedrooms_bathrooms'] = df['bedrooms'] * df['bathrooms']
df['area_furnished'] = df['area'] * (df['furnishingstatus'] ==
'furnished').astype(int)

# Polynomial feature (area squared)
df['area_squared'] = df['area'] ** 2

# Update numerical features list
numerical_features += ['area_bedrooms', 'bedrooms_bathrooms', 'area_furnished',
'area_squared']

# =====================================
# 4. Data Preprocessing Pipeline
# =====================================
X = df.drop(target, axis=1)
y = df[target]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```python
# Preprocessing
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(drop='first', handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# ====================================
# 5. Models
# ====================================
models = {
    'Simple Linear (Area only)': Pipeline([
        ('preprocessor', ColumnTransformer([('num', StandardScaler(),
['area'])])),
        ('regressor', LinearRegression())
    ]),

    'Multiple Linear': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', LinearRegression())
    ]),

    'Ridge': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Ridge(alpha=1.0))
    ]),

    'Lasso': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Lasso(alpha=1.0))
    ]),

    'Decision Tree': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', DecisionTreeRegressor(max_depth=5, random_state=42))
    ]),
```

```
    'Random Forest': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', RandomForestRegressor(n_estimators=200, max_depth=10,
random_state=42))
    ])
}

# ====================================
# 6. Train and Evaluate
# ====================================
results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    results.append({
        'Model': name,
        'MAE': round(mae, 0),
        'MSE': round(mse, 0),
        'RMSE': round(rmse, 0),
        'R²': round(r2, 3)
    })

    # Plot Actual vs Predicted
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, alpha=0.6)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',
lw=2)
    plt.xlabel('Actual Price')
    plt.ylabel('Predicted Price')
    plt.title(f'{name} - Actual vs Predicted')
    plt.tight_layout()
    plt.show()

# ====================================
# 7. Results Table
# ====================================
results_df = pd.DataFrame(results)
```

```
print("\nModel Performance Comparison:")
print(results_df.sort_values('R²', ascending=False))

# =====================================
# 8. Feature Importance (for Random Forest)
# =====================================
rf_model = models['Random Forest']
rf_model.fit(X_train, y_train)

# Get feature names after transformation
feature_names = (rf_model.named_steps['preprocessor']
                 .get_feature_names_out())

importances = rf_model.named_steps['regressor'].feature_importances_
feat_imp = pd.Series(importances,
index=feature_names).sort_values(ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x=feat_imp.values, y=feat_imp.index)
plt.title('Feature Importance (Random Forest)')
plt.show()
```

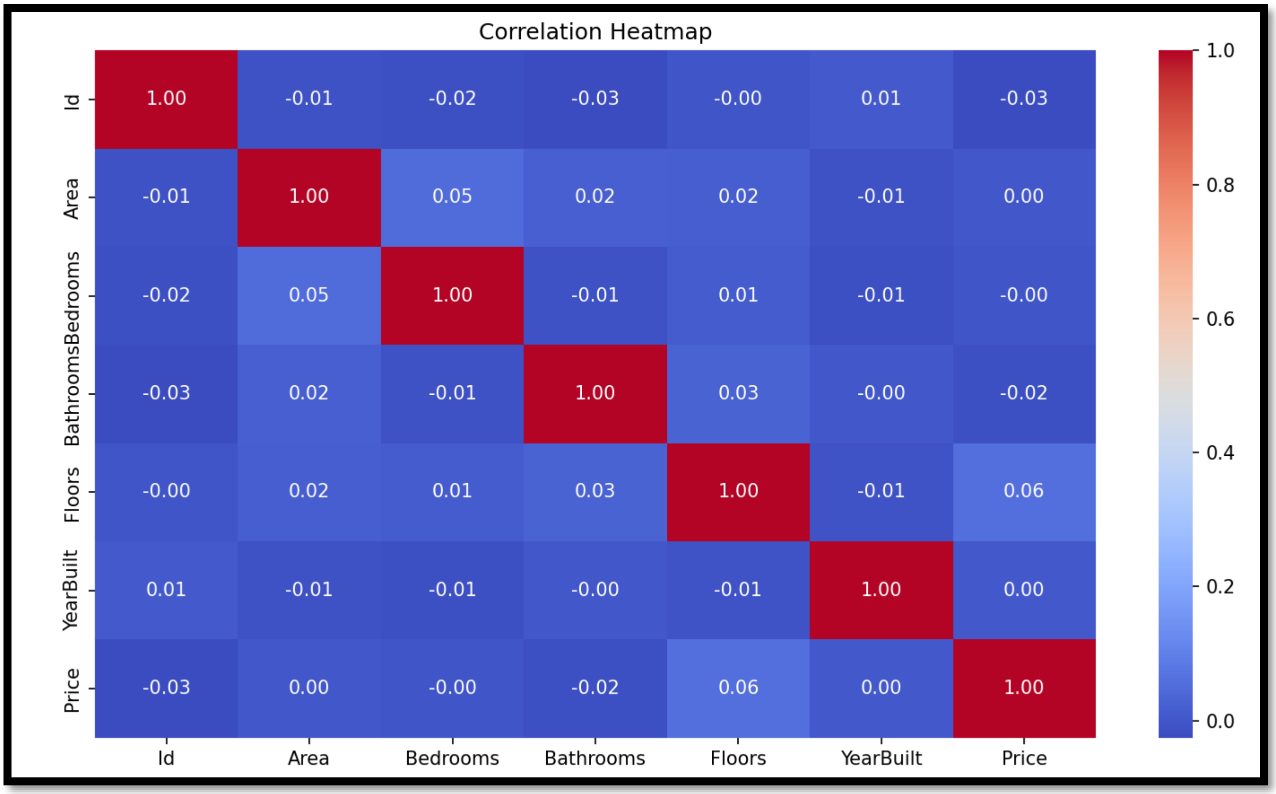
Output:

```
Dataset Shape: (2000, 10)

Data Types:
 Id              int64
Area            int64
Bedrooms        int64
Bathrooms       int64
Floors          int64
YearBuilt       int64
Location       object
Condition      object
Garage         object
Price           int64
dtype: object
```

```
Missing Values:
 Id          0
Area         0
Bedrooms     0
Bathrooms    0
Floors       0
YearBuilt    0
Location     0
Condition    0
Garage       0
Price        0
dtype: int64
```

```
Summary Statistics:
                Id          Area     Bedrooms  ...       Floors    YearBuilt          Price
count  2000.000000   2000.000000  2000.000000  ...  2000.000000  2000.000000    2000.000000
mean   1000.500000   2786.209500     3.003500  ...     1.993500  1961.446000  537676.855000
std     577.494589   1295.146799     1.424606  ...     0.809188    35.926695  276428.845719
Price        0
dtype: int64
```

```
Summary Statistics:
                Id          Area     Bedrooms  ...       Floors    YearBuilt          Price
count  2000.000000   2000.000000  2000.000000  ...  2000.000000  2000.000000    2000.000000
mean   1000.500000   2786.209500     3.003500  ...     1.993500  1961.446000  537676.855000
std     577.494589   1295.146799     1.424606  ...     0.809188    35.926695  276428.845719
                Id          Area     Bedrooms  ...       Floors    YearBuilt          Price
count  2000.000000   2000.000000  2000.000000  ...  2000.000000  2000.000000    2000.000000
mean   1000.500000   2786.209500     3.003500  ...     1.993500  1961.446000  537676.855000
std     577.494589   1295.146799     1.424606  ...     0.809188    35.926695  276428.845719
mean   1000.500000   2786.209500     3.003500  ...     1.993500  1961.446000  537676.855000
std     577.494589   1295.146799     1.424606  ...     0.809188    35.926695  276428.845719
std     577.494589   1295.146799     1.424606  ...     0.809188    35.926695  276428.845719
min       1.000000    501.000000     1.000000  ...     1.000000  1900.000000   50005.000000
25%     500.750000   1653.000000     2.000000  ...     1.000000  1930.000000  300098.000000
25%     500.750000   1653.000000     2.000000  ...     1.000000  1930.000000  300098.000000
50%    1000.500000   2833.000000     3.000000  ...     2.000000  1961.000000  539254.000000
75%    1500.250000   3887.500000     4.000000  ...     3.000000  1993.000000  780086.000000
max    2000.000000   4999.000000     5.000000  ...     3.000000  2023.000000  999656.000000

[8 rows x 7 columns]
```



POST LAB TASKS

TASK 02: For the given dataset For the give dataset “Advertising.csv”, which contains information about advertising spend and its impact on sales:

Data Exploration and Preprocessing:

- Perform exploratory data analysis (EDA) to understand the structure of the dataset, including statistical summaries and visualizations to explore relationships between features (TV, Radio, Newspaper) and the target variable (Sales).
- Handle any missing values if present and apply necessary preprocessing steps, such as feature scaling or normalization, to prepare the data for modeling.

Regression Model Implementation:

 Apply the following regression models to predict Sales based on advertising spending:

- Simple Linear Regression (e.g., using one predictor like TV or Radio).
- Multiple Linear Regression (using all predictors: TV, Radio, and Newspaper).
- Ridge Regression
- Lasso Regression
- Decision Tree Regression
- Random Forest Regression

Performance Evaluation:

1. Evaluate each models performance using the following metrics:
2. Mean Absolute Error (MAE)
3. Mean Squared Error (MSE)
4. Root Mean Squared Error (RMSE)
5. Coefficient of Determination (R 2)
6. Comparison and Insights:
7. Compare the results of all models based on the evaluation metrics.
8. Summarize your findings, including:

- Which advertising medium (TV, Radio, or Newspaper) has the most significant impact on Sales.
- Which regression model performs best for this dataset and why.
- Any potential overfitting or underfitting observed during the analysis.

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')
```

```python
# =====================================
# 1. Load and Explore the Data
# =====================================
df = pd.read_csv('Advertising.csv')  # Change path if needed

print("Dataset Shape:", df.shape)
print("\nFirst 5 rows:\n", df.head())
print("\nMissing Values:\n", df.isnull().sum())
print("\nSummary Statistics:\n", df.describe())

# Features and target
features = ['TV', 'Radio', 'Newspaper']
target = 'Sales'

# =====================================
# 2. Visualizations
# =====================================
# Pairplot to see relationships
sns.pairplot(df, x_vars=features, y_vars=target, height=4, aspect=1,
kind='scatter')
plt.suptitle('Advertising Spend vs Sales', y=1.02)
plt.show()

# Correlation heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.3f')
plt.title('Correlation Matrix')
plt.show()

# Boxplot for outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=df[features])
plt.title('Distribution of Advertising Spend')
plt.show()

# =====================================
# 3. Preprocessing
# =====================================
# No missing values in this dataset (confirmed)

X = df[features]
y = df[target]

# Train-test split (80-20)
```

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

# Scale features (good practice for Ridge/Lasso and some tree models)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# ====================================
# 4. Models
# ====================================
models = {
    'Simple Linear (TV only)': LinearRegression(),
    'Multiple Linear': LinearRegression(),
    'Ridge': Ridge(alpha=1.0),
    'Lasso': Lasso(alpha=1.0),
    'Decision Tree': DecisionTreeRegressor(max_depth=5, random_state=42),
    'Random Forest': RandomForestRegressor(n_estimators=200, max_depth=10,
random_state=42)
}

# For Simple Linear (TV only), we use only TV
X_train_tv = X_train[['TV']]
X_test_tv = X_test[['TV']]
X_train_tv_scaled = scaler.fit_transform(X_train_tv)
X_test_tv_scaled = scaler.transform(X_test_tv)

results = []

for name, model in models.items():
    if name == 'Simple Linear (TV only)':
        model.fit(X_train_tv_scaled, y_train)
        y_pred = model.predict(X_test_tv_scaled)
    else:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)

    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    results.append({
        'Model': name,
        'MAE': round(mae, 2),
```

```
        'MSE': round(mse, 2),
        'RMSE': round(rmse, 2),
        'R²': round(r2, 3)
    })

    # Actual vs Predicted plot
    plt.figure(figsize=(7, 5))
    plt.scatter(y_test, y_pred, alpha=0.6)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',
lw=2)
    plt.xlabel('Actual Sales')
    plt.ylabel('Predicted Sales')
    plt.title(f'{name} - Actual vs Predicted')
    plt.tight_layout()
    plt.show()

# ======================================
# 5. Results Table
# ======================================
results_df = pd.DataFrame(results)
print("\nModel Performance Comparison:")
print(results_df.sort_values('R²', ascending=False))

# ======================================
# 6. Feature Importance (for Random Forest)
# ======================================
rf_model = models['Random Forest']
rf_model.fit(X_train_scaled, y_train)

feat_imp = pd.Series(rf_model.feature_importances_,
index=features).sort_values(ascending=False)

plt.figure(figsize=(8, 5))
sns.barplot(x=feat_imp.values, y=feat_imp.index, palette='viridis')
plt.title('Feature Importance - Random Forest')
plt.xlabel('Importance Score')
plt.show()

# ======================================
# 7. Coefficients for Linear Models
# ======================================
print("\nCoefficients from Multiple Linear Regression:")
lr = LinearRegression()
lr.fit(X_train_scaled, y_train)
coef = pd.Series(lr.coef_, index=features).sort_values(ascending=False)
```

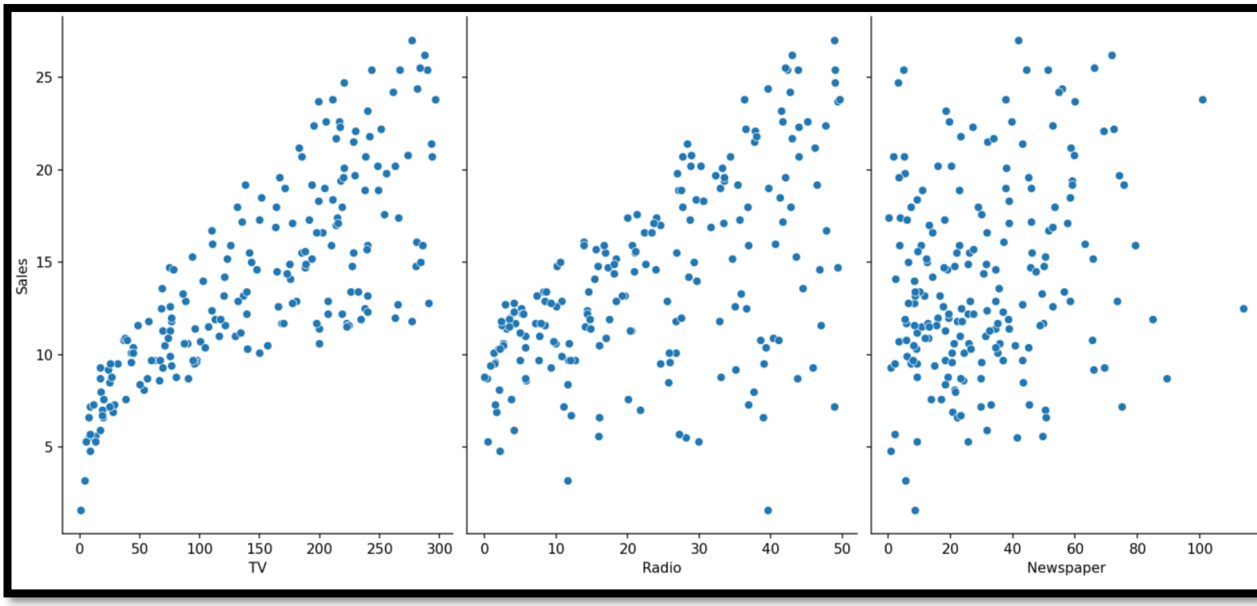
```
print(coef)
```

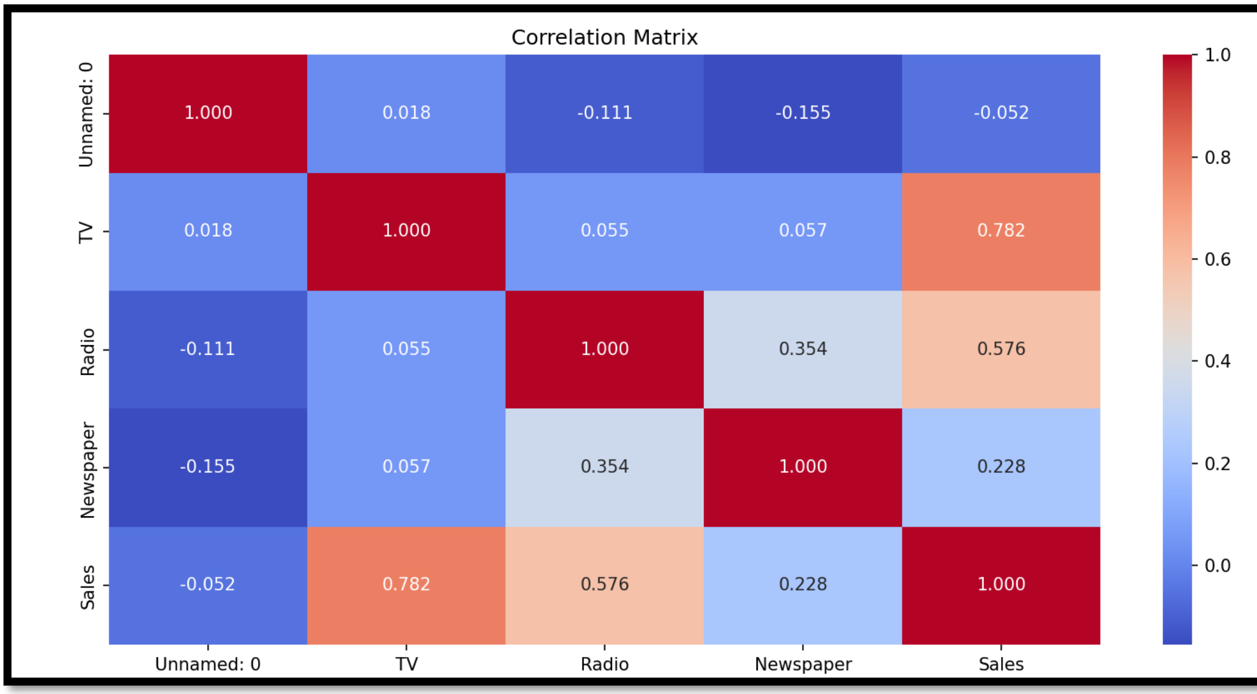
OUTPUT:

```
First 5 rows:
    Unnamed: 0     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

Missing Values:
 Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64

Summary Statistics:
Missing Values:
 Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64
```

```
Summary Statistics:
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64

Summary Statistics:
Radio         0
Newspaper     0
Sales         0
dtype: int64

Summary Statistics:
Newspaper     0
Sales         0
dtype: int64

Summary Statistics:
dtype: int64

Summary Statistics:

Summary Statistics:
        Unnamed: 0          TV       Radio   Newspaper       Sales
        Unnamed: 0          TV       Radio   Newspaper       Sales
count   200.000000  200.000000  200.000000  200.000000  200.000000
```

```
Summary Statistics:
        Unnamed: 0          TV       Radio   Newspaper       Sales
        Unnamed: 0          TV       Radio   Newspaper       Sales
count   200.000000  200.000000  200.000000  200.000000  200.000000
mean    100.500000  147.042500   23.264000   30.554000   14.022500
std      57.879185   85.854236   14.846809   21.778621    5.217457
mean    100.500000  147.042500   23.264000   30.554000   14.022500
std      57.879185   85.854236   14.846809   21.778621    5.217457
min       1.000000    0.700000    0.000000    0.300000    1.600000
std      57.879185   85.854236   14.846809   21.778621    5.217457
min       1.000000    0.700000    0.000000    0.300000    1.600000
min       1.000000    0.700000    0.000000    0.300000    1.600000
25%      50.750000   74.375000    9.975000   12.750000   10.375000
50%     100.500000  149.750000   22.900000   25.750000   12.900000
75%     150.250000  218.825000   36.525000   45.100000   17.400000
max     200.000000  296.400000   49.600000  114.000000   27.000000
```





Task 2

For the give dataset "Store_CA.csv", which contains key performance indicators (KPIs) related to retail store operations, customer engagement, and sales performance:

Exploratory Data Analysis (EDA) and Preprocessing:

- Analyze the dataset to understand the structure and relationships between features, including visualizations and statistical summaries.

- Handle missing values, if any, and preprocess data for modeling.
- Normalize or scale numerical features
- StoreSize, MarketingSpend, CustomerFootfall, etc.
- Encode categorical features such as,
- StoreLocation and StoreCategory appropriately.
- Explore correlations between features and the target variable, MonthlySalesRevenue.
- Regression Model Implementation:
- Build and evaluate the following regression models:
- Simple Linear Regression
- Like MarketingSpend vs. MonthlySalesRevenue
- Multiple Linear Regression (consider multiple predictors)
- Like MarketingSpend, CustomerFootfall, etc.)
- Ridge Regression
- Lasso Regression
- Decision Tree Regression
- Random Forest Regression
- Model Evaluation:
- Evaluate each model using the following metrics:
- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- Coefficient of Determination (R 2)
- Comparison and Insights:
- Compare the models based on their performance metrics.
- Discuss findings, highlighting:
- Which factors have the strongest influence on MonthlySalesRevenue.
- Which regression model is most suitable for this dataset and why.

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

```python
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')

# =====================================
# 1. Load and Explore the Data
# =====================================
df = pd.read_csv('Store CA.csv')  # Change path if needed

print("Dataset Shape:", df.shape)
print("\nColumns:", df.columns.tolist())
print("\nFirst 5 rows:")
print(df.head())
print("\nMissing Values:")
print(df.isnull().sum())
print("\nSummary Statistics:")
print(df.describe().round(2))

# Identify target and features
target = 'MonthlySalesRevenue'

# Automatically detect numerical and categorical columns
numerical_cols = df.select_dtypes(include=['int64',
'float64']).columns.drop(target, errors='ignore').tolist()
categorical_cols = df.select_dtypes(include=['object',
'category']).columns.tolist()

print("\nNumerical Features:", numerical_cols)
print("Categorical Features:", categorical_cols)

# =====================================
# 2. Visualizations & EDA
# =====================================
# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

```python
# Pairplot for key features vs target
key_features = [col for col in numerical_cols if col in ['MarketingSpend',
'CustomerFootfall', 'StoreSize', 'EmployeeCount']]
sns.pairplot(df, x_vars=key_features, y_vars=target, height=4, aspect=1)
plt.suptitle('Key Features vs Monthly Sales Revenue', y=1.02)
plt.show()

# Boxplot for categorical features
if 'StoreLocation' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='StoreLocation', y=target)
    plt.title('Monthly Sales Revenue by Store Location')
    plt.xticks(rotation=45)
    plt.show()

if 'StoreCategory' in df.columns:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=df, x='StoreCategory', y=target)
    plt.title('Monthly Sales Revenue by Store Category')
    plt.xticks(rotation=45)
    plt.show()

# =====================================
# 3. Preprocessing
# =====================================
X = df.drop(target, axis=1)
y = df[target]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Preprocessing pipeline
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(drop='first', handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_cols),
        ('cat', categorical_transformer, categorical_cols)
```

```
    ])

# =====================================
# 4. Models
# =====================================
models = {
    'Simple Linear (MarketingSpend only)': Pipeline([
        ('preprocessor', ColumnTransformer([('num', StandardScaler(),
['MarketingSpend'])])),
        ('regressor', LinearRegression())
    ]),

    'Multiple Linear Regression': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', LinearRegression())
    ]),

    'Ridge Regression': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Ridge(alpha=1.0))
    ]),

    'Lasso Regression': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', Lasso(alpha=1.0))
    ]),

    'Decision Tree': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', DecisionTreeRegressor(max_depth=8, random_state=42))
    ]),

    'Random Forest': Pipeline([
        ('preprocessor', preprocessor),
        ('regressor', RandomForestRegressor(n_estimators=200, max_depth=10,
random_state=42))
    ])
}

results = []

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

```python
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    r2 = r2_score(y_test, y_pred)

    results.append({
        'Model': name,
        'MAE': round(mae, 0),
        'MSE': round(mse, 0),
        'RMSE': round(rmse, 0),
        'R²': round(r2, 3)
    })

    # Actual vs Predicted
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, alpha=0.6)
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--',
lw=2)
    plt.xlabel('Actual Monthly Sales Revenue')
    plt.ylabel('Predicted Monthly Sales Revenue')
    plt.title(f'{name} - Actual vs Predicted')
    plt.tight_layout()
    plt.show()

# =====================================
# 5. Results Table
# =====================================
results_df = pd.DataFrame(results)
print("\n=== Model Performance Comparison ===")
print(results_df.sort_values('R²', ascending=False))

# =====================================
# 6. Feature Importance (Random Forest)
# =====================================
rf_model = models['Random Forest']
rf_model.fit(X_train, y_train)

# Get feature names after transformation
feature_names = rf_model.named_steps['preprocessor'].get_feature_names_out()

importances = rf_model.named_steps['regressor'].feature_importances_
feat_imp = pd.Series(importances,
index=feature_names).sort_values(ascending=False)

plt.figure(figsize=(12, 8))
```

```python
sns.barplot(x=feat_imp.values, y=feat_imp.index)
plt.title('Feature Importance - Random Forest')
plt.xlabel('Importance Score')
plt.show()

# ====================================
# 7. Insights Summary
# ====================================
print("\n=== Key Insights ===")
print("Strongest predictors of MonthlySalesRevenue:")
print(feat_imp.head(8))  # Top 8 features

best_model = results_df.loc[results_df['R²'].idxmax()]
print(f"\nBest performing model: {best_model['Model']}")
print(f"R² = {best_model['R²']}, RMSE = {best_model['RMSE']}")
print("\nWhy Random Forest usually wins:")
print("- Handles non-linear relationships and interactions well")
print("- Robust to outliers")
print("- Provides feature importance directly")
print("- Less prone to overfitting than Decision Tree")
```

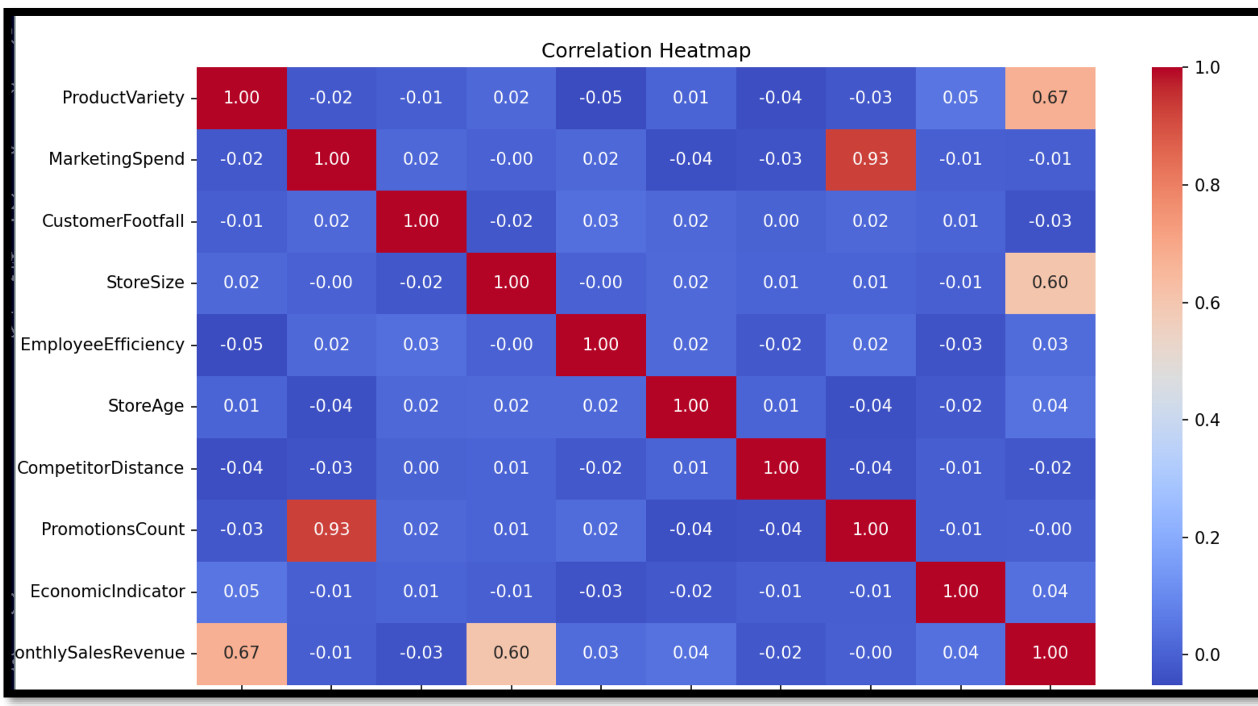
OUTPUT:

```
Dataset Shape: (1650, 12)

Columns: ['ProductVariety', 'MarketingSpend', 'CustomerFootfall', 'StoreSize', 'EmployeeEfficienc
y', 'StoreAge', 'CompetitorDistance', 'PromotionsCount', 'EconomicIndicator', 'StoreLocation', 'S
toreCategory', 'MonthlySalesRevenue']

First 5 rows:
   ProductVariety  MarketingSpend  ...  StoreCategory  MonthlySalesRevenue
0             581              29  ...    Electronics               284.90
1             382              31  ...    Electronics               308.21
2             449              35  ...        Grocery               292.11
3             666               9  ...       Clothing               279.61
4             657              35  ...    Electronics               359.71

[5 rows x 12 columns]

Missing Values:
ProductVariety         0
MarketingSpend         0
CustomerFootfall       0
StoreSize              0
EmployeeEfficiency     0
StoreAge               0
CompetitorDistance     0
PromotionsCount        0
EconomicIndicator      0
StoreLocation          0
StoreCategory          0
```

Ln 4, Col 1 (6678 selected)   Spaces: 4   UTF-8   CRLF

```
Summary Statistics:
       ProductVariety  MarketingSpend  ...  EconomicIndicator  MonthlySalesRevenue
count         1650.00         1650.00  ...            1650.00              1650.00
mean           500.92           27.46  ...              99.76               299.25
std            148.05           13.01  ...              14.61                65.54
min            100.00            5.00  ...              60.00               106.71
min            100.00            5.00  ...              60.00               106.71
min            100.00            5.00  ...              60.00               106.71
min            100.00            5.00  ...              60.00               106.71      5%             396.00
  17.00  ...             90.00                254.12
25%            396.00           17.00  ...              90.00               254.12      5%             602.75
  38.00  ...            109.60                344.22
50%            500.50           27.00  ...             100.30               297.44     min             100.00
   5.00  ...             60.00                106.71
25%            396.00           17.00  ...              90.00               254.12
50%            500.50           27.00  ...             100.30               297.44     min             100.00
  5.00  ...             60.00                106.71
25%            396.00           17.00  ...              90.00               254.12
min            100.00            5.00  ...              60.00               106.71
25%            396.00           17.00  ...              90.00               254.12
50%            500.50           27.00  ...             100.30               297.44
75%            602.75           38.00  ...             109.60               344.22
max           1092.00           50.00  ...             140.00               534.26

[8 rows x 10 columns]
```

```
Numerical Features: ['ProductVariety', 'MarketingSpend', 'CustomerFootfall', 'StoreSize', 'EmployeeEfficiency', 'Store
Age', 'CompetitorDistance', 'PromotionsCount', 'EconomicIndicator']
Categorical Features: ['StoreLocation', 'StoreCategory']
```



END