```
1
```

```
1 from google.colab import drive
2 drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import numpy as np
```

```
1 data = pd.read_csv('/content/drive/MyDrive/AirQualityUCI.csv')
```

```
1 data.head()
```

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(
0	3/10/2004	18:00:00	2.6	1360	150	11.9	1046	
1	3/10/2004	19:00:00	2.0	1292	112	9.4	955	
2	3/10/2004	20:00:00	2.2	1402	88	9.0	939	
3	3/10/2004	21:00:00	2.2	1376	80	9.2	948	
4	3/10/2004	22:00:00	1.6	1272	51	6.5	836	

```
1 data.columns=['DATE','TIME','CO_GT','PT08_S1_CO','NMHC_GT','C6H6_GT','PT08_S2_NMHC','NOX_GT','PT08_S3_NOX','NO2_GT','PT08_S4_NO2','P
```

```
1 data.shape
```

```
(9357, 15)
```

```
1 data.head()
```

	DATE	TIME	CO_GT	PT08_S1_CO	NMHC_GT	C6H6_GT	PT08_S2_NMHC	NOX_GT	P
0	3/10/2004	18:00:00	2.6	1360	150	11.9	1046	166	
1	3/10/2004	19:00:00	2.0	1292	112	9.4	955	103	
2	3/10/2004	20:00:00	2.2	1402	88	9.0	939	131	
3	3/10/2004	21:00:00	2.2	1376	80	9.2	948	172	
4	3/10/2004	22:00:00	1.6	1272	51	6.5	836	131	

```
1 data.replace(to_replace= -200, value= np.NaN, inplace= True)
```

```
1 data.isna().sum()
```

```
DATE              0
TIME              0
CO_GT          1683
PT08_S1_CO      366
NMHC_GT        8443
C6H6_GT         366
PT08_S2_NMHC    366
NOX_GT         1639
PT08_S3_NOX     366
NO2_GT         1642
PT08_S4_NO2     366
PT08_S5_O3      366
T               366
RH              366
AH              366
dtype: int64
```

```
1 data.drop('NMHC_GT',axis=1,inplace=True)
```

```
1 data.head()
```

	DATE	TIME	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	NOX_GT	PT08_S3_NO
0	3/10/2004	18:00:00	2.6	1360.0	11.9	1046.0	166.0	1056.
1	3/10/2004	19:00:00	2.0	1292.0	9.4	955.0	103.0	1174.
2	3/10/2004	20:00:00	2.2	1402.0	9.0	939.0	131.0	1140.

```
1 data.isna().sum()
```

```
DATE              0
TIME              0
CO_GT          1683
PT08_S1_CO      366
C6H6_GT         366
PT08_S2_NMHC    366
NOX_GT         1639
PT08_S3_NOX     366
NO2_GT         1642
PT08_S4_NO2     366
PT08_S5_O3      366
T               366
RH              366
AH              366
dtype: int64
```

```
1 data.dtypes
```

```
DATE            object
TIME            object
CO_GT          float64
PT08_S1_CO     float64
C6H6_GT        float64
PT08_S2_NMHC   float64
NOX_GT         float64
PT08_S3_NOX    float64
NO2_GT         float64
PT08_S4_NO2    float64
PT08_S5_O3     float64
T              float64
RH             float64
AH             float64
dtype: object
```

```
1 data.describe()
```

	CO_GT	PT08_S1_CO	C6H6_GT	PT08_S2_NMHC	NOX_GT	PT08_S3_NO
count	7674.000000	8991.000000	8991.000000	8991.000000	7718.000000	8991.00000(
mean	2.152750	1099.833166	10.083105	939.153376	246.896735	835.49360
std	1.453252	217.080037	7.449820	266.831429	212.979168	256.81732(
min	0.100000	647.000000	0.100000	383.000000	2.000000	322.00000(
25%	1.100000	937.000000	4.400000	734.500000	98.000000	658.00000(
50%	1.800000	1063.000000	8.200000	909.000000	180.000000	806.00000(
75%	2.900000	1231.000000	14.000000	1116.000000	326.000000	969.50000(
max	11.900000	2040.000000	63.700000	2214.000000	1479.000000	2683.00000(

```
1 data.dropna().shape
```

```
(6941, 14)
```

```
1 data.dropna(inplace=True)
```

```
1 data.isnull().sum()
```

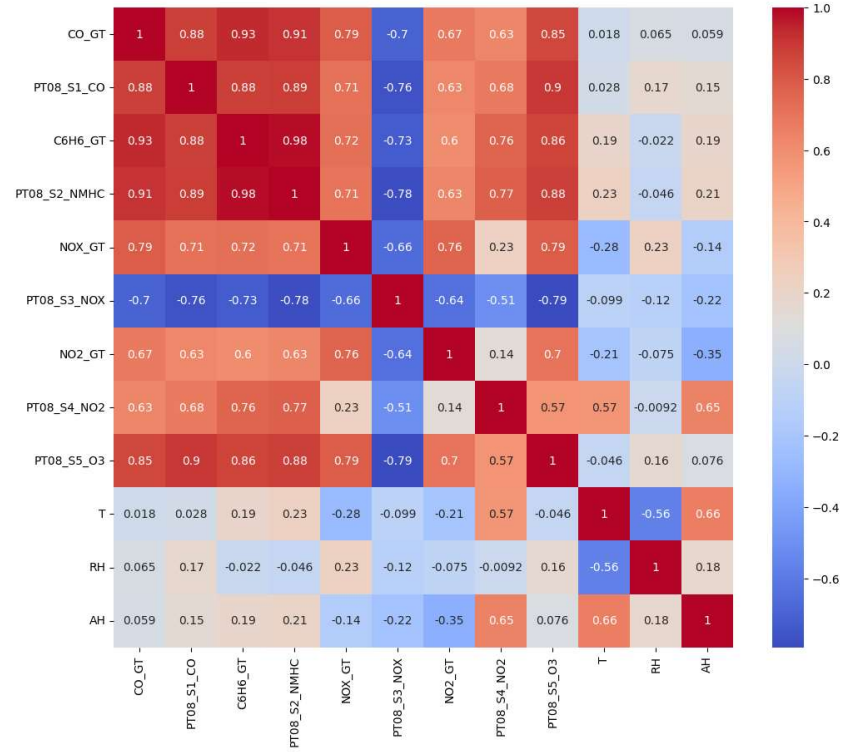
```
DATE           0
TIME           0
CO_GT          0
PT08_S1_CO     0
C6H6_GT        0
PT08_S2_NMHC   0
NOX_GT         0
PT08_S3_NOX    0
NO2_GT         0
PT08_S4_NO2    0
PT08_S5_O3     0
T              0
RH             0
```

```
AH             0
dtype: int64
```

Data Visualization

```
correlation_matrix = data.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.show()
```

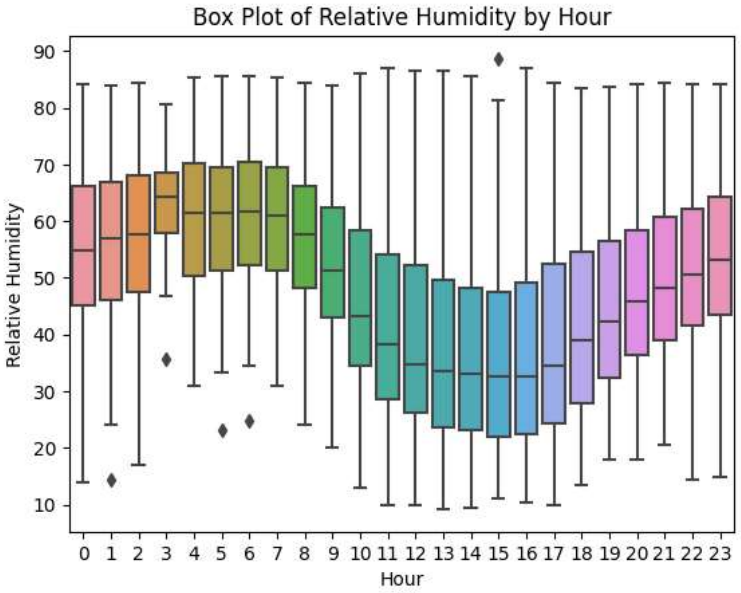
```
<ipython-input-19-6c76563d8704>:1: FutureWarning: The default value of numeric_onl
  correlation_matrix = data.corr()
```



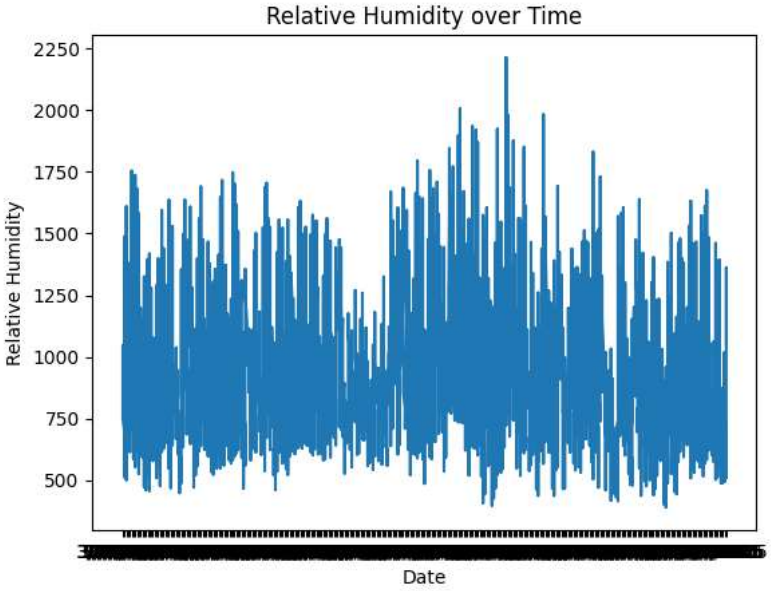

```
data['HOUR']=data['TIME'].apply(lambda x: int(x.split(':')[0]))
data.HOUR
```

```
0       18
1       19
2       20
3       21
4       22
        ..
9352    10
9353    11
9354    12
9355    13
9356    14
Name: HOUR, Length: 6941, dtype: int64
```

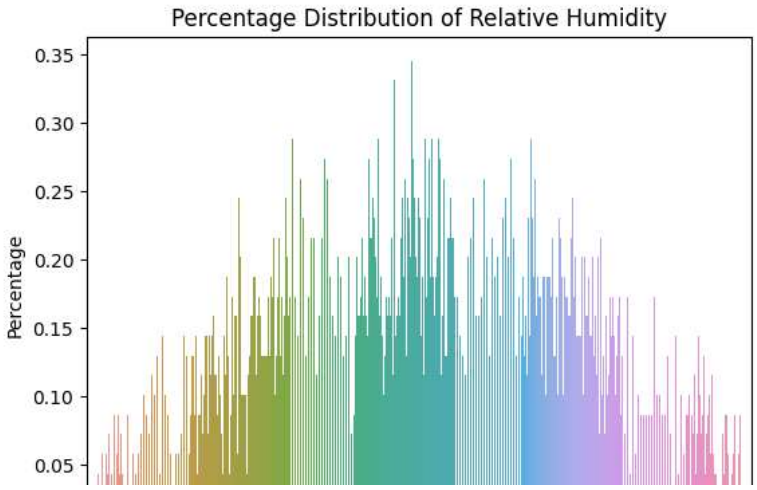
```
sns.boxplot(x='HOUR', y='RH', data=data)
plt.xlabel('Hour')
plt.ylabel('Relative Humidity')
plt.title('Box Plot of Relative Humidity by Hour')
plt.show()

```




```
plt.plot(data['DATE'], data['PT08_S2_NMHC'])
plt.xlabel('Date')
plt.ylabel('Relative Humidity')
plt.title('Relative Humidity over Time')
plt.show()

```




```
# Import the necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Plotting a bar plot of RH
sns.barplot(data=data, x='RH', y='RH', estimator=lambda x: len(x) / len(data) * 100)
plt.xlabel('Relative Humidity')
plt.ylabel('Percentage')
plt.title('Percentage Distribution of Relative Humidity')
plt.show()

```

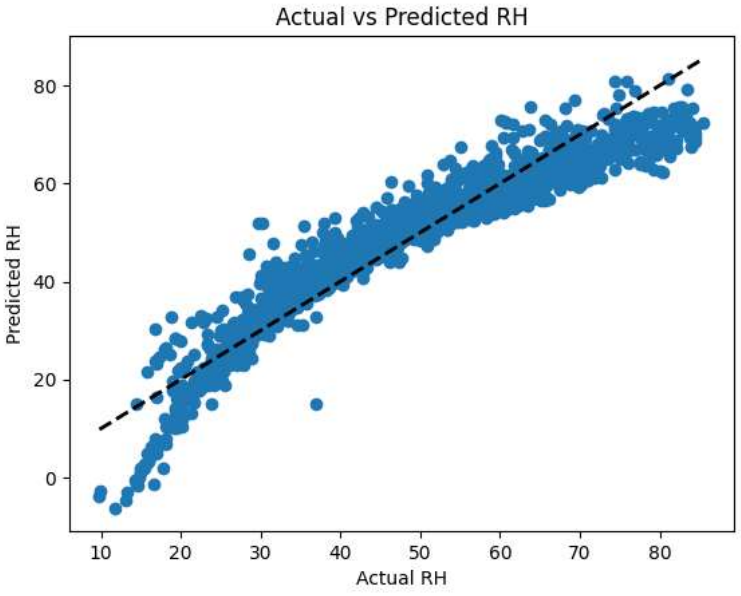


Linear Regression

```
1 from sklearn.model_selection import train_test_split      #import train test split
2 from sklearn.linear_model import LinearRegression         #import linear regression package
3 from sklearn.metrics import mean_squared_error,mean_absolute_error   #import mean squared error and mean absolute error
4 from sklearn.metrics import r2_score
```

```
1 X = data.drop(['RH','DATE','TIME'], axis=1)
2 y = data['RH']
3
4 # Split the data into training and testing sets
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
6
7 # Create and train the linear regression model
8 linear_regression = LinearRegression()
9 linear_regression.fit(X_train, y_train)
10
11 # Make predictions on the test set
12 y_pred_lr = linear_regression.predict(X_test)
```

```
1 # Plotting actual vs predicted values
2 plt.scatter(y_test, y_pred_lr)
3 plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
4 plt.xlabel('Actual RH')
5 plt.ylabel('Predicted RH')
6 plt.title('Actual vs Predicted RH')
7 plt.show()
```




```
1 mean_squared_error(y_test, y_pred_lr)
```

```
34.0802484967582
```

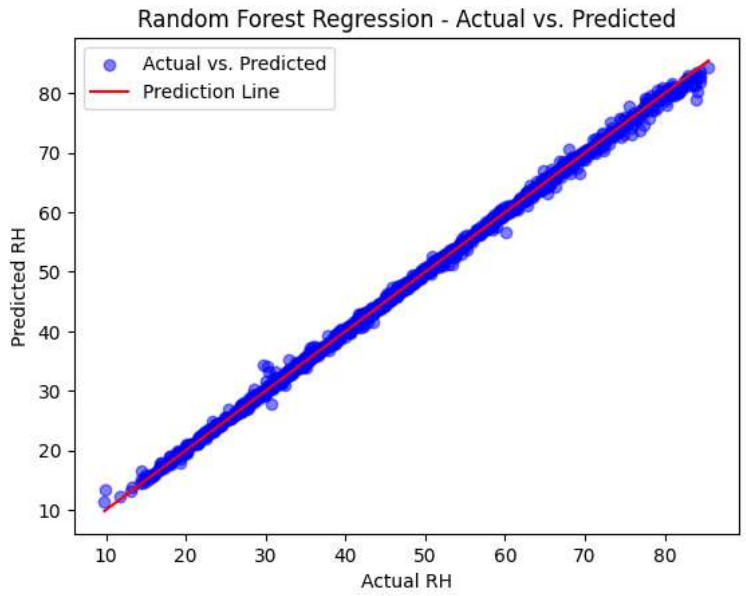
```
r2_score(y_test, y_pred_lr)
```

```
0.8899362175811059
```

Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
random_forest_model = RandomForestRegressor()
random_forest_model.fit(X_train, y_train)
y_pred_rf = random_forest_model.predict(X_test)
```

```
plt.scatter(y_test, y_pred_rf, color='blue', label='Actual vs. Predicted',alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', label='Prediction Line')
plt.xlabel('Actual RH')
plt.ylabel('Predicted RH')
plt.title('Random Forest Regression - Actual vs. Predicted')
plt.legend()
plt.show()
```




```
r2_score(y_test, y_pred_rf)
```

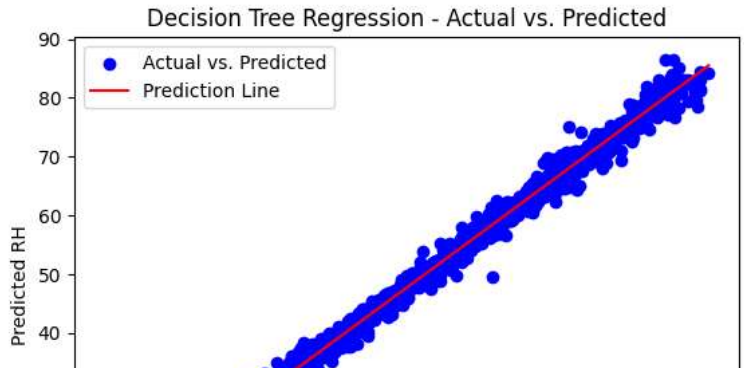
```
0.9985749187561072
```

Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
```

```
decision_tree_model = DecisionTreeRegressor()
decision_tree_model.fit(X_train, y_train)
y_pred_dt = decision_tree_model.predict(X_test)
```

```
plt.scatter(y_test, y_pred_dt, color='blue', label='Actual vs. Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', label='Prediction Line')
plt.xlabel('Actual RH')
plt.ylabel('Predicted RH')
plt.title('Decision Tree Regression - Actual vs. Predicted')
plt.legend()
plt.show()
```




```
1 mean_squared_error(y_test, y_pred_rf)
```

```
0.4412634370050409
```

```
1 r2_score(y_test, y_pred_dt)
```

```
0.9940341491503423
```

SVM

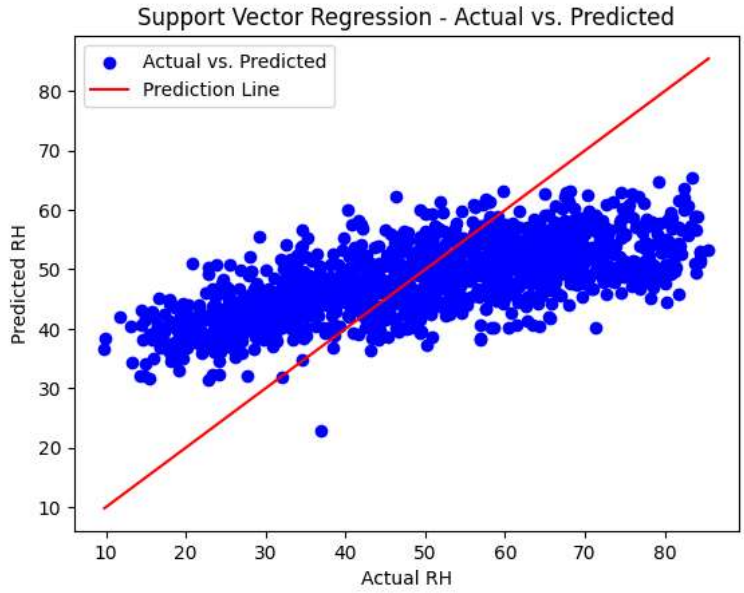
```
1 from sklearn.svm import SVR
```

```
1 svr_model = SVR()
2 svr_model.fit(X_train, y_train)
3 y_pred_svr = svr_model.predict(X_test)
```

```
1 r2_score(y_test, y_pred_svr)
```

```
0.3681904342526975
```

```
1 plt.scatter(y_test, y_pred_svr, color='blue', label='Actual vs. Predicted')
2 plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', label='Prediction Line')
3 plt.xlabel('Actual RH')
4 plt.ylabel('Predicted RH')
5 plt.title('Support Vector Regression - Actual vs. Predicted')
6 plt.legend()
7 plt.show()
```



Lasso Regression

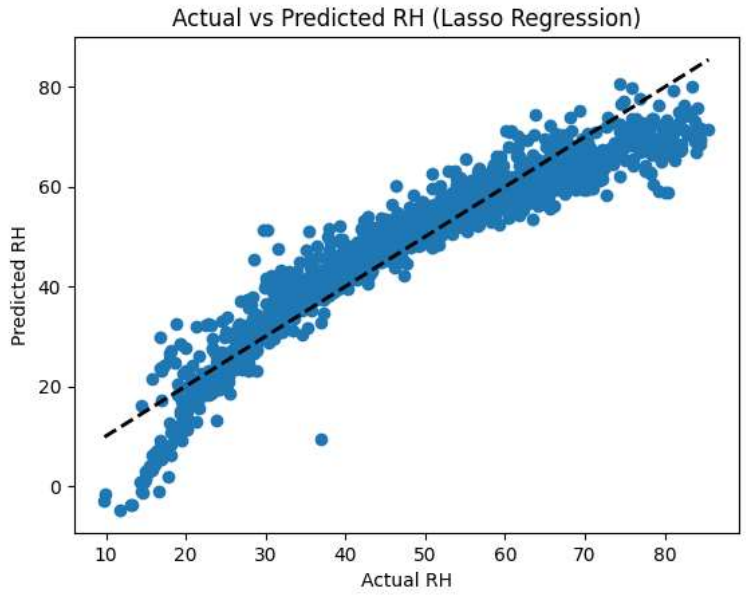
```
1  from sklearn.linear_model import Lasso
2  lasso_model = Lasso(alpha=0.1)  # You can adjust the value of alpha
3  lasso_model.fit(X_train, y_train)
4
```

```
# Make predictions on the test set
y_pred_lm = lasso_model.predict(X_test)
```

```
r2_score(y_test, y_pred_lm)
```

```
0.8878969070776364
```

```
plt.scatter(y_test, y_pred_lm)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual RH')
plt.ylabel('Predicted RH')
plt.title('Actual vs Predicted RH (Lasso Regression)')
plt.show()
```



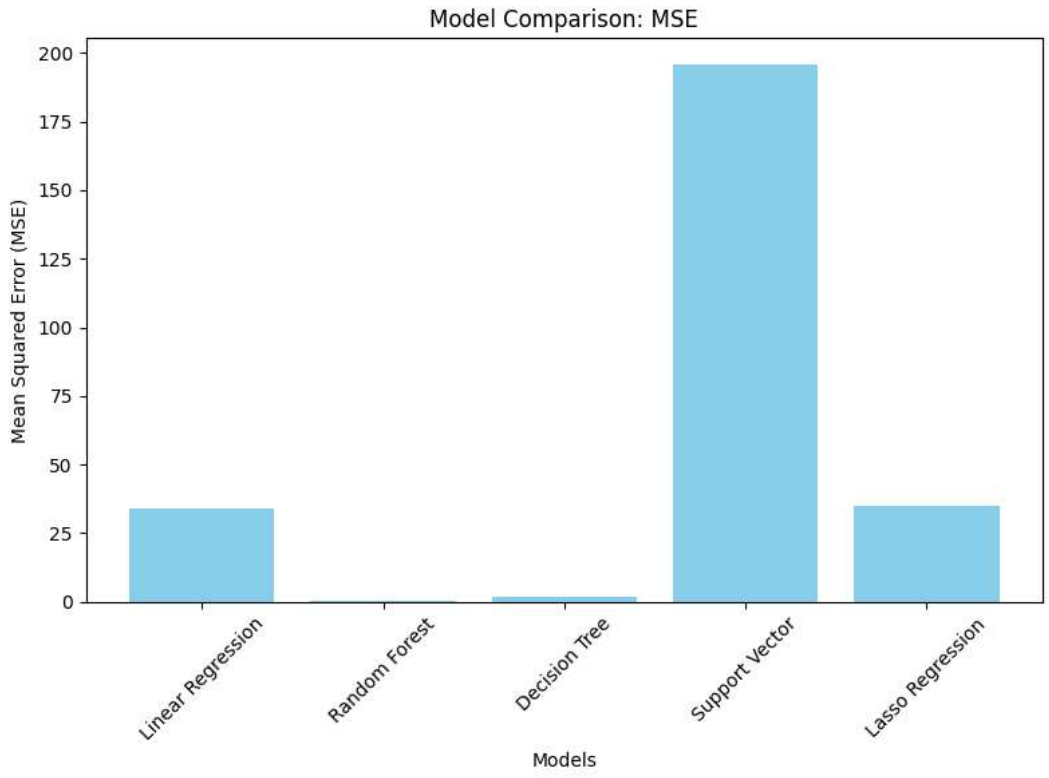
Comparison

```
from sklearn.metrics import mean_squared_error, r2_score

# Calculate MSE for each models
mse_lr = mean_squared_error(y_test, y_pred_lr)
mse_rf = mean_squared_error(y_test, y_pred_rf)
mse_dt = mean_squared_error(y_test, y_pred_dt)
mse_svr = mean_squared_error(y_test, y_pred_svr)
mse_lm = mean_squared_error(y_test, y_pred_lm)

# Calculate R2 score for each model
r2_lr = r2_score(y_test, y_pred_lr)
r2_rf = r2_score(y_test, y_pred_rf)
r2_dt = r2_score(y_test, y_pred_dt)
r2_svr = r2_score(y_test, y_pred_svr)
r2_lm = r2_score(y_test, y_pred_lm)

# Create a dataframe to compare the metrics
metrics_df = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest', 'Decision Tree', 'SVR', 'Lasso Regression'],
    'MSE': [mse_lr, mse_rf, mse_dt, mse_svr, mse_lm],
    'R2 Score': [r2_lr, r2_rf, r2_dt, r2_svr, r2_lm]
})

# Display the dataframe
print(metrics_df)
```

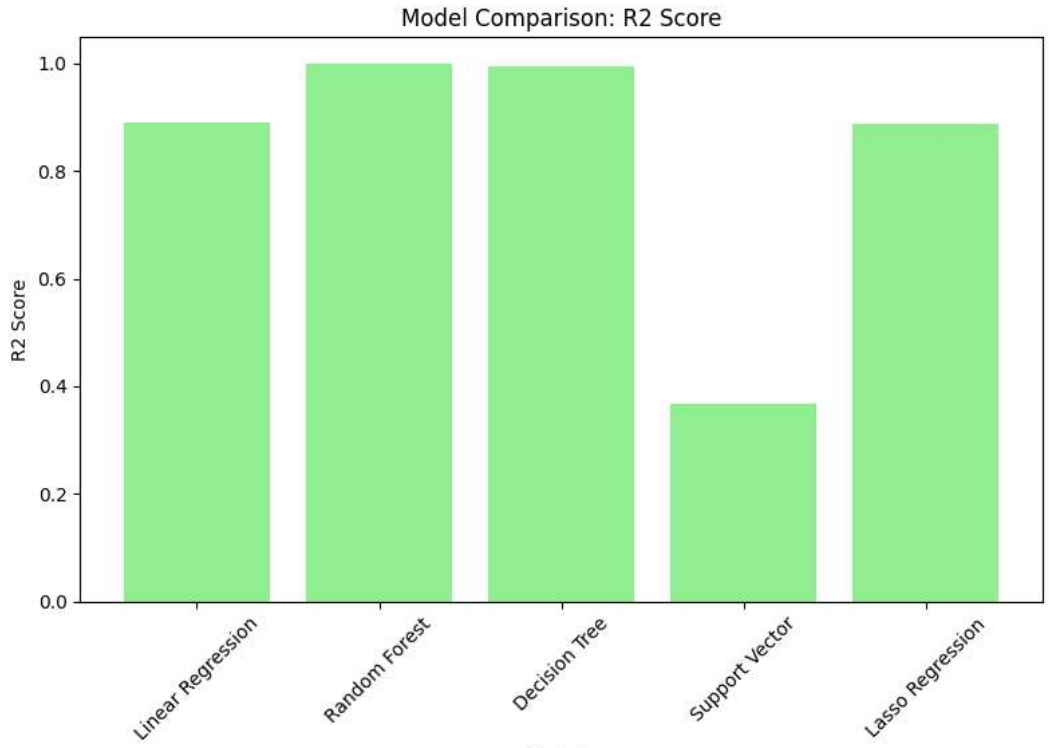
```
                Model         MSE  R2 Score
0   Linear Regression   34.080248  0.889936
1       Random Forest    0.441263  0.998575
2       Decision Tree    1.847271  0.994034
3                 SVR  195.634082  0.368190
4    Lasso Regression   34.711702  0.887897
```

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Create lists of models, MSE values, and R2 scores
models = ['Linear Regression', 'Random Forest', 'Decision Tree', 'Support Vector', 'Lasso Regression']
mse_values = [mse_lr, mse_rf, mse_dt, mse_svr, mse_lm]
r2_scores = [r2_lr, r2_rf, r2_dt, r2_svr, r2_lm]

# Plot MSE comparison
plt.figure(figsize=(8, 6))
plt.bar(models, mse_values, color='skyblue')
plt.xlabel('Models')
plt.ylabel('Mean Squared Error (MSE)')
plt.title('Model Comparison: MSE')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()


```




```
# Plot R2 score comparison
plt.figure(figsize=(8, 6))
plt.bar(models, r2_scores, color='lightgreen')
plt.xlabel('Models')
plt.ylabel('R2 Score')
plt.title('Model Comparison: R2 Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```



✓ 0s completed at 10:09 PM