Ayan_Sur_002410504003_Assignment_2

Question 1. Write Python code to implement a regression model for weather forecasting. Compare simple regression, Ridge regression and Lasso regression

***THEORY :***

**Regression Techniques :**

1. *Linear Regression*

Goal: Finds the best-fitting line (or hyperplane in higher dimensions) through the data points by minimizing the sum of squared errorsbetween the predicted and actual values. Equation:

[ y = mx + c ]

y : Target variable

x : Feature variable

m : Slope of the line

c : Intercept

Pros: Simple, interpretable, and widely used.

Cons: Prone to overfitting if there are many features or if the features are highly correlated (multicollinearity).

2. *Ridge Regression*

Goal: Similar to Linear Regression but adds a penalty term (L2 regularization) to the cost function to shrink the coeffi cients towards zero.

Pros: Reduces overfi tting by constraining the size of the coeffi cients.Handles multicollinearity better than Linear Regression.

Cons: Does not perform feature selection (all features are kept in the model).

3. *Lasso Regression*

Goal: Similar to Ridge Regression but uses a different penalty term (L1 regularization) that can shrink some coeffi cients to exactly zero.

Pros: Performs feature selection by eliminating less important features. Handles multicollinearity well.

Cons: Can be unstable when features are highly correlated and might arbitrarily select one feature over another.

```
# Importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Load the dataset
from google.colab import files
uploaded = files.upload()
data = pd.read_csv(list(uploaded.keys())[0])
print(data.head())

# Explore the dataset
print("Dataset Head:")
print(data.head())
print("\nDataset Info:")
print(data.info())
print("\nMissing Values:\n", data.isnull().sum())

# Preprocessing: Handle missing values if necessary (e.g., fill or drop)
data = data.dropna()  # Dropping rows with missing values (if any)


# Separating features and target variable
# Assuming "Temperature" is the target and the rest are features
X = data.drop(columns=['tem'])
y = data['tem']
```

```
# Splitting the Training and Testing Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize models
linear_model = LinearRegression()
ridge = Ridge()
lasso = Lasso(max_iter=10000)

# Hyperparameter tuning for Ridge and Lasso
ridge_params = {'alpha': [0.01, 0.1, 1, 10, 100]}
lasso_params = {'alpha': [0.01, 0.1, 1, 10, 100]}

ridge_search = GridSearchCV(ridge, param_grid=ridge_params, scoring='neg_mean_squared_error', cv=5)
lasso_search = GridSearchCV(lasso, param_grid=lasso_params, scoring='neg_mean_squared_error', cv=5)

# Train and evaluate models
models = {
    'Linear Regression': linear_model,
    'Ridge Regression': ridge_search,
    'Lasso Regression': lasso_search
}

results = {}
for name, model in models.items():
    if name == 'Linear Regression':
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
    else:
        model.fit(X_train_scaled, y_train)
        y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results[name] = {'MSE': mse, 'R2': r2}
    print(f"{name}:")
    print(f" Mean Squared Error: {mse:.2f}")
    print(f" R-squared: {r2:.2f}\n")

# Visualization of predictions
plt.figure(figsize=(12, 6))
markers = {'Linear Regression': 'o', 'Ridge Regression': 's', 'Lasso Regression': 'x'}
colors = {'Linear Regression': 'blue', 'Ridge Regression': 'orange', 'Lasso Regression': 'green'}

for name, model in models.items():
    if name == 'Linear Regression':
        y_pred = model.predict(X_test)
    else:
        y_pred = model.predict(X_test_scaled)

    # Plot predictions
    plt.scatter(y_test, y_pred, label=name, alpha=0.7, marker=markers[name], color=colors[name])

# Plot the perfect prediction line
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, label="Perfect Prediction")

# Set labels, legend, and title
plt.xlabel("Actual Temperature")
plt.ylabel("Predicted Temperature")
plt.legend()
plt.title("Actual vs Predicted Temperatures")
plt.show()
```

Choose Files Temp_and_rain.csv

- **Temp_and_rain.csv**(text/csv) - 33189 bytes, last modified: 11/21/2024 - 100% done

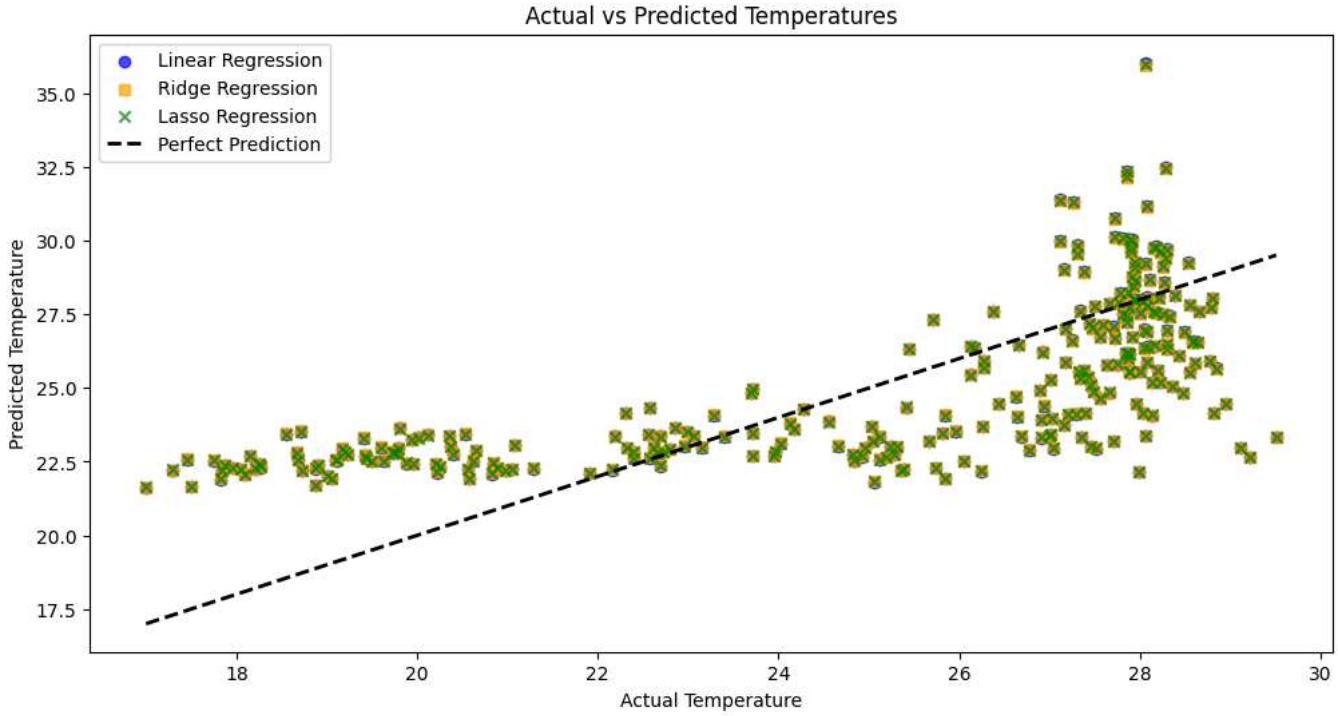
```
Saving Temp_and_rain.csv to Temp_and_rain (1).csv
       tem  Month  Year      rain
0  16.9760      1  1901   18.5356
1  19.9026      2  1901   16.2548
2  24.3158      3  1901   70.7981
3  28.1834      4  1901   66.1616
4  27.8892      5  1901  267.2150
Dataset Head:
       tem  Month  Year      rain
0  16.9760      1  1901   18.5356
1  19.9026      2  1901   16.2548
2  24.3158      3  1901   70.7981
3  28.1834      4  1901   66.1616
4  27.8892      5  1901  267.2150

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1380 entries, 0 to 1379
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   tem     1380 non-null   float64
 1   Month   1380 non-null   int64
 2   Year    1380 non-null   int64
 3   rain    1380 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 43.2 KB
None

Missing Values:
 tem      0
Month    0
Year     0
rain     0
dtype: int64
Linear Regression:
 Mean Squared Error: 6.90
 R-squared: 0.45

Ridge Regression:
 Mean Squared Error: 6.88
 R-squared: 0.45

Lasso Regression:
 Mean Squared Error: 6.88
 R-squared: 0.45
```




***Explanation :***

This code trains three different regression models (Linear Regression, Ridge Regression, and Lasso Regression) to predict weather (specifically, temperature) and then evaluates their performance using two key metrics:

Mean Squared Error (MSE): This measures the average squared difference between the actual temperature values and the values predicted by the models. A lower MSE indicates better accuracy.

R-squared (R2): This represents the proportion of the variance in the target variable (temperature) that is explained by the model. It ranges from 0 to 1, with higher values indicating a better fit.

The code then prints the MSE and R2 for each model, allowing to compare their performance. Additionally, it generates a scatter plot that visually shows how the predicted temperatures from each model compare to the actual temperatures.

*Why Results Differ*:

The differences in performance between Linear Regression, Ridge Regression, and Lasso Regression arise from the way each model handles the relationships between features (predictors) and the target variable.

**Linear Regression**: This model tries to find the best-fitting straight line through the data points. It doesn't impose any constraints on the model's coefficients (the weights assigned to each feature). This can lead to overfitting if there are many features or if some features are highly correlated. Overfitting means the model performs well on the training data but poorly on unseen data.

**Ridge Regression**: This model introduces a penalty term to the cost function that is used to train the model. This penalty term discourages large coefficients, effectively shrinking them towards zero. This helps to prevent overfitting and makes the model more robust to multicollinearity (high correlation between features).

**Lasso Regression**: Like Ridge Regression, Lasso Regression also uses a penalty term to shrink coefficients. However, Lasso can shrink some coefficients all the way to zero, effectively performing feature selection. This can be helpful in identifying the most important features for predicting the target variable.

In Summary,

*Linear Regression* is the most basic model and can be prone to **Overfitting**.

*Ridge Regression* addresses overfitting by shrinking coefficients and is good for handling multicollinearity.

*Lasso Regression* can perform feature selection by shrinking some coefficients to zero.

The specific differences noticed in the output (MSE and R2 values) for weather forecasting task depend on the nature of the data and the relationships between the features and the target variable. Moreover one model performs better than others for given particular dataset. Experimenting with different models and hyperparameters is often necessary to achieve the best results.