```python
import numpy as np;
import pandas as pd
import matplotlib.pyplot as plt;
import seaborn as sn

#load the dataset
dataset=pd.read_csv("c:\\Users\\lsrin\\Downloads\\TS-2\\Adavance_ML\\
climate.csv")
print(dataset)
```

```
          STATION                 DATE REPORT_TYPE  SOURCE
BackupElements  \
0     72518014735  2015-01-01T23:59:00       SOD          6
PRECIP
1     72518014735  2015-01-02T23:59:00       SOD          6
PRECIP
2     72518014735  2015-01-03T23:59:00       SOD          6
PRECIP
3     72518014735  2015-01-04T23:59:00       SOD          6
PRECIP
4     72518014735  2015-01-05T23:59:00       SOD          6
PRECIP
...           ...                  ...         ...     ...
...
2663  72518014735  2022-05-27T23:59:00       SOD          6
PRECIP
2664  72518014735  2022-05-28T23:59:00       SOD          6
PRECIP
2665  72518014735  2022-05-29T23:59:00       SOD          6
PRECIP
2666  72518014735  2022-05-30T23:59:00       SOD          6
PRECIP
2667  72518014735  2022-05-31T23:59:00       SOD          6
PRECIP

      BackupElevation BackupEquipment  BackupLatitude  BackupLongitude
\
0                 260         PLASTIC         42.6918        -73.83109

1                 260         PLASTIC         42.6918        -73.83109

2                 260         PLASTIC         42.6918        -73.83109

3                 260         PLASTIC         42.6918        -73.83109

4                 260         PLASTIC         42.6918        -73.83109

...               ...             ...             ...              ...

2663              260         PLASTIC         42.6812        -73.81650
```

```
2664              260        PLASTIC        42.6812        -73.81650

2665              260        PLASTIC        42.6812        -73.81650

2666              260        PLASTIC        42.6812        -73.81650

2667              260        PLASTIC        42.6812        -73.81650


          BackupName  ...  DailyPeakWindDirection  DailyPeakWindSpeed  \
0     NWS ALBANY, NY  ...                   190.0                26.0

1     NWS ALBANY, NY  ...                   250.0                30.0

2     NWS ALBANY, NY  ...                   170.0                21.0

3     NWS ALBANY, NY  ...                   290.0                33.0

4     NWS ALBANY, NY  ...                   280.0                42.0

...              ...  ...                     ...                 ...

2663  NWS ALBANY, NY  ...                   160.0                28.0

2664  NWS ALBANY, NY  ...                   310.0                26.0

2665  NWS ALBANY, NY  ...                    90.0                13.0

2666  NWS ALBANY, NY  ...                   200.0                15.0

2667  NWS ALBANY, NY  ...                   250.0                29.0


      DailyPrecipitation  DailySnowDepth  DailySnowfall  \
0                   0.00             0.0            0.0
1                      T             0.0              T
2                   0.57             0.0            1.6
3                   0.22             1.0            0.0
4                      T             0.0              T
...                  ...             ...            ...
2663                0.00               0            0.0
2664                0.04               0            0.0
2665                0.00               0            0.0
2666                0.00               0            0.0
2667                0.00               0            0.0

      DailySustainedWindDirection  DailySustainedWindSpeed  Sunrise
Sunset  \
0                           190.0                     20.0    726.0
```

```
1632.0
1                              310.0                       23.0    726.0
1633.0
2                              160.0                       15.0    726.0
1634.0
3                              290.0                       24.0    726.0
1635.0
4                              290.0                       32.0    726.0
1636.0
...                              ...                        ...      ...
...
2663                           160.0                       21.0    423.0
1922.0
2664                           310.0                       22.0    422.0
1923.0
2665                           180.0                        9.0    421.0
1924.0
2666                           190.0                       12.0    421.0
1925.0
2667                           250.0                       21.0    420.0
1926.0

      WindEquipmentChangeDate
0                  2006-09-08
1                  2006-09-08
2                  2006-09-08
3                  2006-09-08
4                  2006-09-08
...                       ...
2663               2006-09-08
2664               2006-09-08
2665               2006-09-08
2666               2006-09-08
2667               2006-09-08

[2668 rows x 32 columns]
```

```
dataset.columns
```

```
Index(['STATION', 'DATE', 'REPORT_TYPE', 'SOURCE', 'BackupElements',
       'BackupElevation', 'BackupEquipment', 'BackupLatitude',
       'BackupLongitude', 'BackupName',
'DailyAverageDewPointTemperature',
       'DailyAverageDryBulbTemperature',
'DailyAverageRelativeHumidity',
       'DailyAverageSeaLevelPressure', 'DailyAverageStationPressure',
       'DailyAverageWetBulbTemperature', 'DailyAverageWindSpeed',
       'DailyCoolingDegreeDays',
'DailyDepartureFromNormalAverageTemperature',
       'DailyHeatingDegreeDays', 'DailyMaximumDryBulbTemperature',
```

```
       'DailyMinimumDryBulbTemperature', 'DailyPeakWindDirection',
       'DailyPeakWindSpeed', 'DailyPrecipitation', 'DailySnowDepth',
       'DailySnowfall', 'DailySustainedWindDirection',
       'DailySustainedWindSpeed', 'Sunrise', 'Sunset',
       'WindEquipmentChangeDate'],
      dtype='object')
```

```
#DATA PROCESSSING
dataset.head(10)
dataset.tail(8)
dataset.sample(7)
print(dataset.columns)
print(dataset.dtypes)
print(dataset.shape[0]) #column
print(dataset.shape[1]) #rows
print(dataset.size)
```

```
Index(['STATION', 'DATE', 'REPORT_TYPE', 'SOURCE', 'BackupElements',
       'BackupElevation', 'BackupEquipment', 'BackupLatitude',
       'BackupLongitude', 'BackupName',
'DailyAverageDewPointTemperature',
       'DailyAverageDryBulbTemperature',
'DailyAverageRelativeHumidity',
       'DailyAverageSeaLevelPressure', 'DailyAverageStationPressure',
       'DailyAverageWetBulbTemperature', 'DailyAverageWindSpeed',
       'DailyCoolingDegreeDays',
'DailyDepartureFromNormalAverageTemperature',
       'DailyHeatingDegreeDays', 'DailyMaximumDryBulbTemperature',
       'DailyMinimumDryBulbTemperature', 'DailyPeakWindDirection',
       'DailyPeakWindSpeed', 'DailyPrecipitation', 'DailySnowDepth',
       'DailySnowfall', 'DailySustainedWindDirection',
       'DailySustainedWindSpeed', 'Sunrise', 'Sunset',
       'WindEquipmentChangeDate'],
      dtype='object')
STATION                                                  int64
DATE                                                    object
REPORT_TYPE                                             object
SOURCE                                                   int64
BackupElements                                          object
BackupElevation                                          int64
BackupEquipment                                         object
BackupLatitude                                         float64
BackupLongitude                                        float64
BackupName                                              object
DailyAverageDewPointTemperature                        float64
DailyAverageDryBulbTemperature                         float64
DailyAverageRelativeHumidity                           float64
DailyAverageSeaLevelPressure                           float64
DailyAverageStationPressure                            float64
DailyAverageWetBulbTemperature                         float64
```

```
DailyAverageWindSpeed                          float64
DailyCoolingDegreeDays                         float64
DailyDepartureFromNormalAverageTemperature    float64
DailyHeatingDegreeDays                         float64
DailyMaximumDryBulbTemperature                 float64
DailyMinimumDryBulbTemperature                 float64
DailyPeakWindDirection                         float64
DailyPeakWindSpeed                             float64
DailyPrecipitation                              object
DailySnowDepth                                  object
DailySnowfall                                   object
DailySustainedWindDirection                    float64
DailySustainedWindSpeed                        float64
Sunrise                                        float64
Sunset                                         float64
WindEquipmentChangeDate                         object
dtype: object
2668
32
85376
```

```
dataset.isnull().sum()
```

```
STATION                                       0
DATE                                          0
REPORT_TYPE                                   0
SOURCE                                        0
BackupElements                                0
BackupElevation                               0
BackupEquipment                               0
BackupLatitude                                0
BackupLongitude                               0
BackupName                                    0
DailyAverageDewPointTemperature               0
DailyAverageDryBulbTemperature                0
DailyAverageRelativeHumidity                  0
DailyAverageSeaLevelPressure                  0
DailyAverageStationPressure                   0
DailyAverageWetBulbTemperature                0
DailyAverageWindSpeed                         0
DailyCoolingDegreeDays                        0
DailyDepartureFromNormalAverageTemperature    0
DailyHeatingDegreeDays                        0
DailyMaximumDryBulbTemperature                0
DailyMinimumDryBulbTemperature                0
DailyPeakWindDirection                        0
DailyPeakWindSpeed                            0
DailyPrecipitation                            0
DailySnowDepth                                0
DailySnowfall                                 0
```

```
DailySustainedWindDirection                        0
DailySustainedWindSpeed                            0
Sunrise                                            0
Sunset                                             0
WindEquipmentChangeDate                            0
dtype: int64
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2668 entries, 0 to 2667
Data columns (total 32 columns):
 #   Column                                      Non-Null Count  Dtype

---  ------                                      --------------  -----

 0   STATION                                     2668 non-null   int64

 1   DATE                                        2668 non-null
object
 2   REPORT_TYPE                                 2668 non-null
object
 3   SOURCE                                      2668 non-null   int64

 4   BackupElements                              2668 non-null
object
 5   BackupElevation                             2668 non-null   int64

 6   BackupEquipment                             2668 non-null
object
 7   BackupLatitude                              2668 non-null
float64
 8   BackupLongitude                             2668 non-null
float64
 9   BackupName                                  2668 non-null
object
 10  DailyAverageDewPointTemperature             2668 non-null
float64
 11  DailyAverageDryBulbTemperature              2668 non-null
float64
 12  DailyAverageRelativeHumidity                2668 non-null
float64
 13  DailyAverageSeaLevelPressure                2668 non-null
float64
 14  DailyAverageStationPressure                 2668 non-null
float64
 15  DailyAverageWetBulbTemperature              2668 non-null
float64
 16  DailyAverageWindSpeed                       2668 non-null
float64
```

```
 17  DailyCoolingDegreeDays                      2668 non-null
float64
 18  DailyDepartureFromNormalAverageTemperature  2668 non-null
float64
 19  DailyHeatingDegreeDays                      2668 non-null
float64
 20  DailyMaximumDryBulbTemperature              2668 non-null
float64
 21  DailyMinimumDryBulbTemperature              2668 non-null
float64
 22  DailyPeakWindDirection                      2668 non-null
float64
 23  DailyPeakWindSpeed                          2668 non-null
float64
 24  DailyPrecipitation                          2668 non-null
object
 25  DailySnowDepth                              2668 non-null
object
 26  DailySnowfall                               2668 non-null
object
 27  DailySustainedWindDirection                 2668 non-null
float64
 28  DailySustainedWindSpeed                     2668 non-null
float64
 29  Sunrise                                     2668 non-null
float64
 30  Sunset                                      2668 non-null
float64
 31  WindEquipmentChangeDate                     2668 non-null
object
dtypes: float64(20), int64(3), object(9)
memory usage: 667.1+ KB

# Ensure 'DailyPrecipitation' is numeric and fill NaN values with 0
dataset['DailyPrecipitation'] =
pd.to_numeric(dataset['DailyPrecipitation'], errors='coerce')
dataset['DailyPrecipitation'].isnull().sum()
dataset['DailyPrecipitation'].fillna(0)

0       0.00
1       0.00
2       0.57
3       0.22
4       0.00
        ...
2663    0.00
2664    0.04
2665    0.00
2666    0.00
```

```
2667    0.00
Name: DailyPrecipitation, Length: 2668, dtype: float64
```

```
dataset['DailyPrecipitation'].isnull().sum()
```

```
0
```

```
#do some statistical for int64,float64
dataNumerical=dataset.select_dtypes(include=['int64','float64'])
#print(dataNumerical.columns)
print(dataNumerical.dtypes)
```

```
STATION                                         int64
SOURCE                                          int64
BackupElevation                                 int64
BackupLatitude                                float64
BackupLongitude                               float64
DailyAverageDewPointTemperature               float64
DailyAverageDryBulbTemperature                float64
DailyAverageRelativeHumidity                  float64
DailyAverageSeaLevelPressure                  float64
DailyAverageStationPressure                   float64
DailyAverageWetBulbTemperature                float64
DailyAverageWindSpeed                         float64
DailyCoolingDegreeDays                        float64
DailyDepartureFromNormalAverageTemperature    float64
DailyHeatingDegreeDays                        float64
DailyMaximumDryBulbTemperature                float64
DailyMinimumDryBulbTemperature                float64
DailyPeakWindDirection                        float64
DailyPeakWindSpeed                            float64
DailyPrecipitation                            float64
DailySustainedWindDirection                   float64
DailySustainedWindSpeed                       float64
Sunrise                                       float64
Sunset                                        float64
dtype: object
```

```
#drop the uncessary attribute
dataNumerical = dataNumerical.drop(columns=[
    'STATION', 'SOURCE', 'BackupElevation', 'BackupLatitude',
    'BackupLongitude', 'Sunrise', 'Sunset'
])
```

```
dataNumerical.columns
```

```
Index(['DailyAverageDewPointTemperature',
'DailyAverageDryBulbTemperature',
       'DailyAverageRelativeHumidity', 'DailyAverageSeaLevelPressure',
       'DailyAverageStationPressure',
'DailyAverageWetBulbTemperature',
```

```
       'DailyAverageWindSpeed', 'DailyCoolingDegreeDays',
       'DailyDepartureFromNormalAverageTemperature',
'DailyHeatingDegreeDays',
       'DailyMaximumDryBulbTemperature',
'DailyMinimumDryBulbTemperature',
       'DailyPeakWindDirection', 'DailyPeakWindSpeed',
'DailyPrecipitation',
       'DailySustainedWindDirection', 'DailySustainedWindSpeed'],
      dtype='object')
```

```
#Using linear Regression
# Features matrix (X) with the temperature column
X = dataNumerical[['DailyAverageDryBulbTemperature']]
# Target vector (y) with precipitation values
y = dataNumerical['DailyPrecipitation']


# Print X and y to verify
print(X)
print(y.isnull().sum())
```

```
      DailyAverageDryBulbTemperature
0                               25.0
1                               32.0
2                               27.0
3                               39.0
4                               27.0
...                              ...
2663                            70.0
2664                            68.0
2665                            66.0
2666                            72.0
2667                            79.0

[2668 rows x 1 columns]
0
```

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=32)

# Print the number of rows in X_train
print(X_train.shape[0])
```

```
2134
```

```
#train the model
from sklearn.linear_model import LinearRegression
# Initialize and train the linear regression model
```

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Make predictions
y_pred = model.predict(X_test)
#print(y_pred)
#y_pred

# Evaluate the model
from sklearn.metrics import mean_squared_error, r2_score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.0729113990218066
R-squared: 0.0025889789634933047
```

2. Feature Selection Techniques(Low Variance Filter)

The goal is to remove features with very low variance,as they don't provide useful information. For example, if a feature value barely changes across data samples, it won't help the model predict precipitation.

S1: Remove the X feature =DailyPrecipitation

s2:VarianceThreshold wit 0.1 which are <0.1 contain low variances s3:fit the selectors

```
from sklearn.feature_selection import VarianceThreshold

# Step 1: Drop the target column (DailyPrecipitation) from the feature matrix
X = dataNumerical.drop(columns=['DailyPrecipitation'])
print("All Features:")
print(X.columns)

# Step 2: Initialize the VarianceThreshold selector with a specific threshold (e.g., 0.1)
selector = VarianceThreshold(threshold=0.1)

# Step 3: Fit the selector on the feature matrix
selector.fit(X)

# Step 4: Calculate variances for each feature
variances = X.var()
print("\nVariance of each feature:")
print(variances)
```

```
# Step 5: Identify threshold value
threshold = selector.threshold
print(f"\nVariance Threshold: {threshold}")

# Step 6: Identify selected and removed features based on the
threshold
selected_columns = variances[variances >= threshold].index
removed_columns = variances[variances < threshold].index

# Step 7: Print selected and removed features
print("\nSelected Features after Low Variance Filter:")
print(selected_columns)

print("\nRemoved Features with Low Variance:")
print(removed_columns)
```

```
All Features:
Index(['DailyAverageDewPointTemperature',
'DailyAverageDryBulbTemperature',
       'DailyAverageRelativeHumidity', 'DailyAverageSeaLevelPressure',
       'DailyAverageStationPressure',
'DailyAverageWetBulbTemperature',
       'DailyAverageWindSpeed', 'DailyCoolingDegreeDays',
       'DailyDepartureFromNormalAverageTemperature',
'DailyHeatingDegreeDays',
       'DailyMaximumDryBulbTemperature',
'DailyMinimumDryBulbTemperature',
       'DailyPeakWindDirection', 'DailyPeakWindSpeed',
       'DailySustainedWindDirection', 'DailySustainedWindSpeed'],
      dtype='object')

Variance of each feature:
DailyAverageDewPointTemperature                 365.431000
DailyAverageDryBulbTemperature                  351.461615
DailyAverageRelativeHumidity                    179.596433
DailyAverageSeaLevelPressure                      0.050074
DailyAverageStationPressure                       0.048773
DailyAverageWetBulbTemperature                  290.359311
DailyAverageWindSpeed                            14.805719
DailyCoolingDegreeDays                           17.913956
DailyDepartureFromNormalAverageTemperature       67.288101
DailyHeatingDegreeDays                          260.312571
DailyMaximumDryBulbTemperature                  400.148238
DailyMinimumDryBulbTemperature                  328.421192
DailyPeakWindDirection                         8249.827997
DailyPeakWindSpeed                               89.043312
DailySustainedWindDirection                    8253.098128
DailySustainedWindSpeed                          48.192936
dtype: float64
```

```
Variance Threshold: 0.1

Selected Features after Low Variance Filter:
Index(['DailyAverageDewPointTemperature',
'DailyAverageDryBulbTemperature',
       'DailyAverageRelativeHumidity',
'DailyAverageWetBulbTemperature',
       'DailyAverageWindSpeed', 'DailyCoolingDegreeDays',
       'DailyDepartureFromNormalAverageTemperature',
'DailyHeatingDegreeDays',
       'DailyMaximumDryBulbTemperature',
'DailyMinimumDryBulbTemperature',
       'DailyPeakWindDirection', 'DailyPeakWindSpeed',
       'DailySustainedWindDirection', 'DailySustainedWindSpeed'],
      dtype='object')

Removed Features with Low Variance:
Index(['DailyAverageSeaLevelPressure', 'DailyAverageStationPressure'],
dtype='object')
```

```
X_selected = dataNumerical[selected_columns]  # This extracts the data
matrix

# Target variable remains the same
y = dataNumerical['DailyPrecipitation']

# Verify shapes of X and y
print(X_selected.shape)
print(y.shape)

# Split the data into train and test sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_selected, y,
test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(2668, 14)
(2668,)
(2134, 14) (534, 14) (2134,) (534,)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)
```

```python
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
r2LVF = r2_score(y_test, y_pred)
mseLVF = mean_squared_error(y_test, y_pred)

print(f'R² Score: {r2LVF:.2f}')
print(f'Mean Squared Error: {mseLVF:.2f}')
```

```
R² Score: 0.29
Mean Squared Error: 0.05
```

These visualizations will give you:

Feature distributions – Helps understand spread and outliers. Scatter plot – Visualizes relationships between features and target. Correlation heatmap – Identifies multicollinearity. Residual plot – Checks for bias in predictions. Line plot – Visualizes how well predictions align with actual values. Bar plot – Summarizes model metrics.

```python
dataNumerical.columns
```

```
Index(['DailyAverageDewPointTemperature',
'DailyAverageDryBulbTemperature',
       'DailyAverageRelativeHumidity', 'DailyAverageSeaLevelPressure',
       'DailyAverageStationPressure',
'DailyAverageWetBulbTemperature',
       'DailyAverageWindSpeed', 'DailyCoolingDegreeDays',
       'DailyDepartureFromNormalAverageTemperature',
'DailyHeatingDegreeDays',
       'DailyMaximumDryBulbTemperature',
'DailyMinimumDryBulbTemperature',
       'DailyPeakWindDirection', 'DailyPeakWindSpeed',
'DailyPrecipitation',
       'DailySustainedWindDirection', 'DailySustainedWindSpeed'],
      dtype='object')
```

Applying a High Correlation Filter (HCF) is a great way to reduce feature dimensionality by removing features that are highly correlated with each other. This helps to prevent multicollinearity issues in your model, which can affect the interpretability and performance of regression models.

```python
X_selected = dataNumerical.drop(columns=['DailyPrecipitation'])  #
Ensure to remove the target

# Step 1: Calculate the correlation matrix
correlation_matrix = X_selected.corr()
```

```python
# Step 2: Set a correlation threshold
threshold = 0.8

# Step 3: Identify features to remove
correlated_features = set()  # Set to store features to remove

# Iterate through the correlation matrix
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > threshold:
            # Get the name of the feature
            colname = correlation_matrix.columns[i]
            correlated_features.add(colname)

# Display correlated features
print("Correlated Features to Remove:")
print(correlated_features)

# Step 4: Drop correlated features from the feature matrix
X_filtered = X_selected.drop(columns=correlated_features)

# Display the remaining features
print("\nFinal Features after High Correlation Filter:")
print(X_filtered.columns)
```

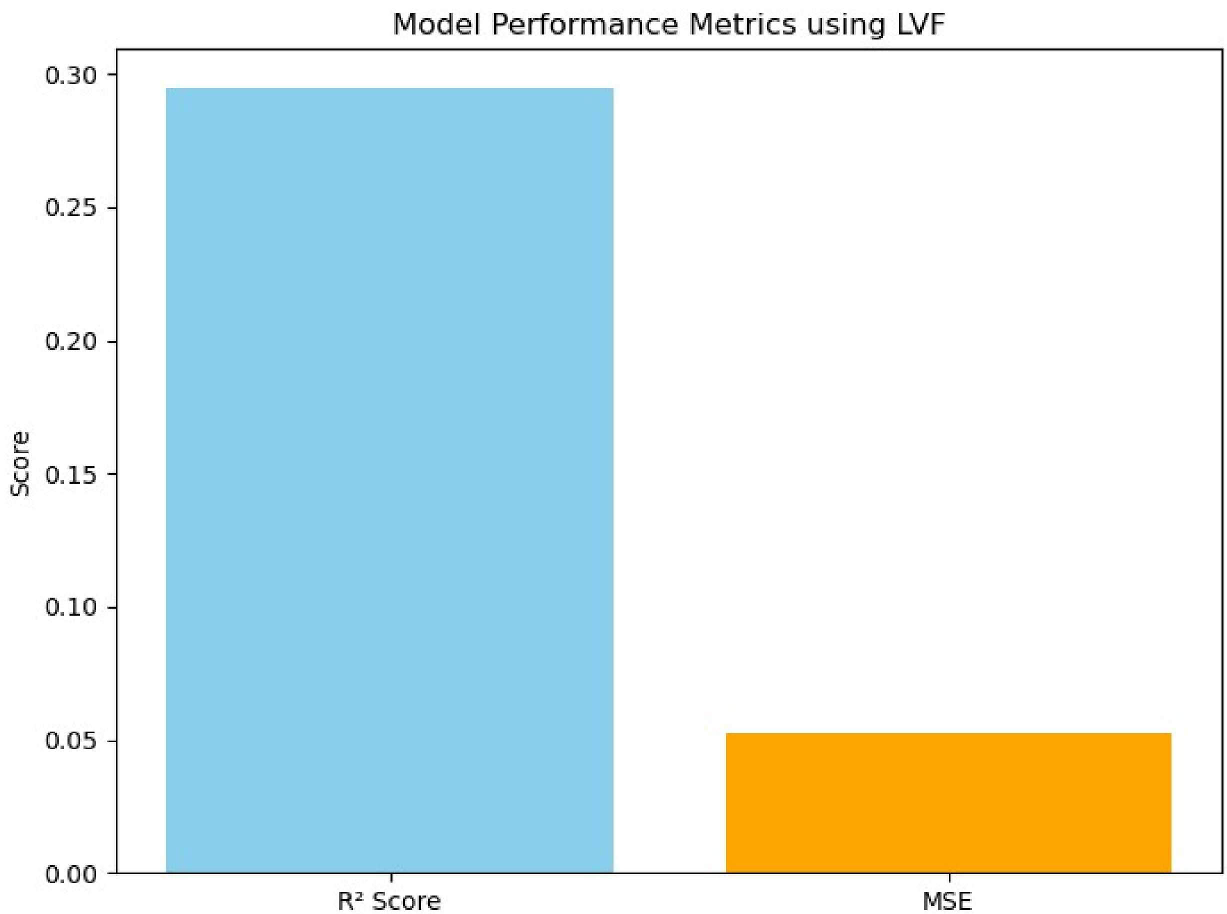
```
Correlated Features to Remove:
{'DailyMinimumDryBulbTemperature', 'DailyAverageWetBulbTemperature',
'DailyAverageStationPressure', 'DailyHeatingDegreeDays',
'DailySustainedWindSpeed', 'DailyAverageDryBulbTemperature',
'DailyMaximumDryBulbTemperature', 'DailyPeakWindSpeed'}

Final Features after High Correlation Filter:
Index(['DailyAverageDewPointTemperature',
'DailyAverageRelativeHumidity',
       'DailyAverageSeaLevelPressure', 'DailyAverageWindSpeed',
       'DailyCoolingDegreeDays',
'DailyDepartureFromNormalAverageTemperature',
       'DailyPeakWindDirection', 'DailySustainedWindDirection'],
      dtype='object')
```

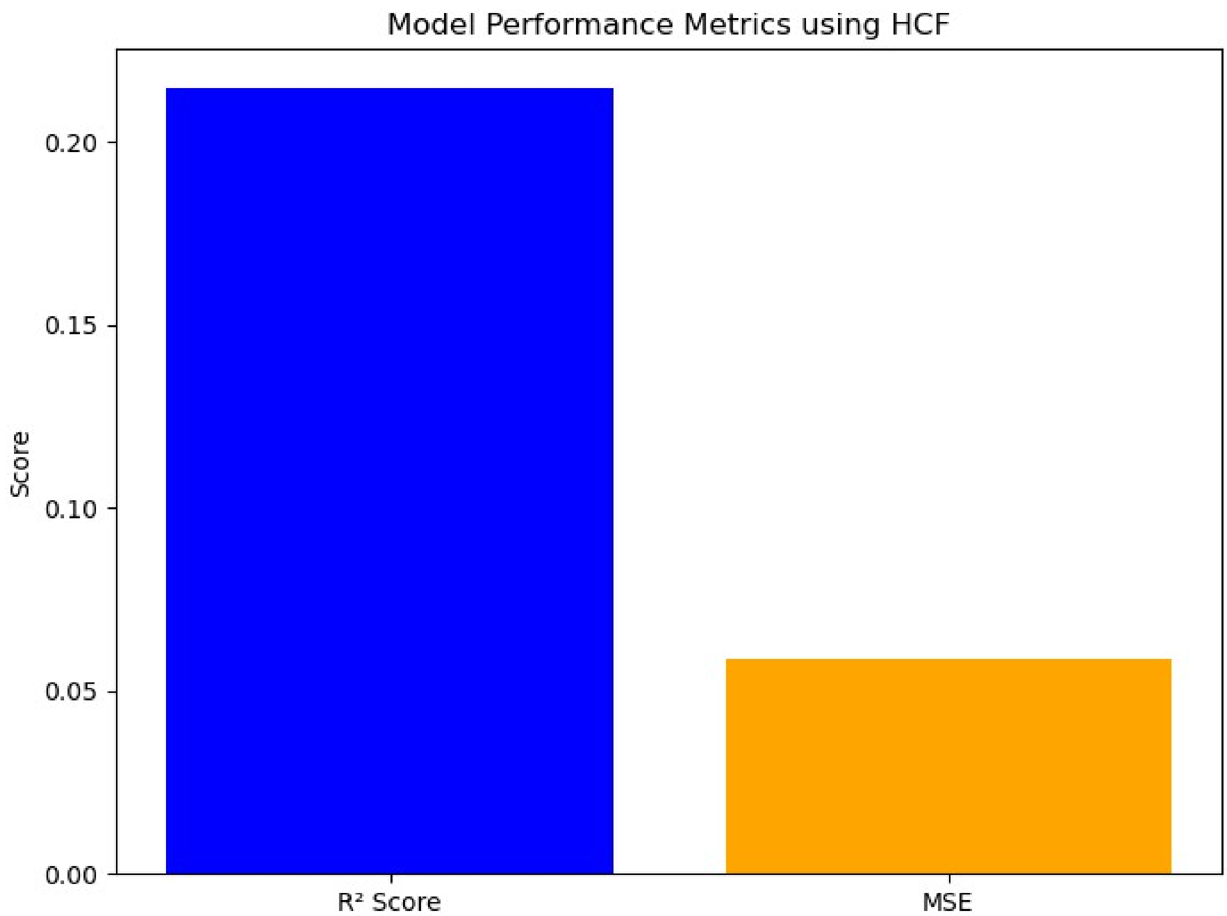
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# Use the filtered features for training and evaluation
y = dataNumerical['DailyPrecipitation']

# Split the filtered data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_filtered, y,
test_size=0.2, random_state=42)
```

```python
# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
r2HCF= r2_score(y_test, y_pred)
msehcf= mean_squared_error(y_test, y_pred)

print(f'R² Score: {r2HCF:.2f}')
print(f'Mean Squared Error: {msehcf:.2f}')
```

```
R² Score: 0.21
Mean Squared Error: 0.06
```

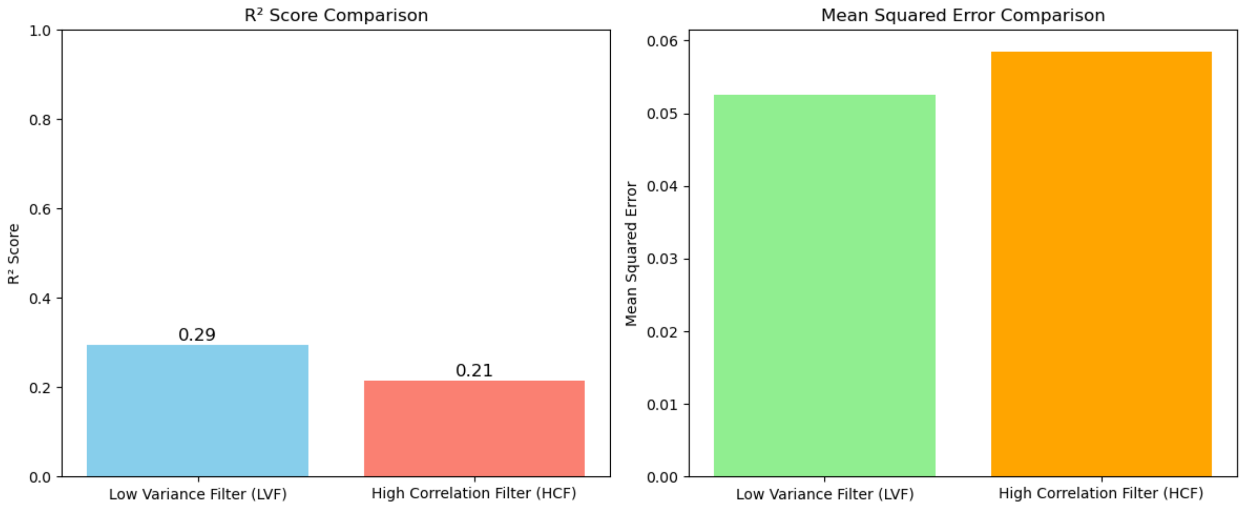
```python
# Example R² Score visualization
metrics = {'R² Score':r2LVF, 'MSE': mseLVF}
plt.figure(figsize=(8, 6))
plt.bar(metrics.keys(), metrics.values(), color=['skyblue', 'orange'])
plt.title("Model Performance Metrics using LVF")
plt.ylabel("Score")
plt.show()
#HCF# Example R² Score visualization
metrics = {'R² Score':r2HCF, 'MSE':msehcf}
plt.figure(figsize=(8, 6))
plt.bar(metrics.keys(), metrics.values(), color=['blue', 'orange'])
plt.title("Model Performance Metrics using HCF")
plt.ylabel("Score")
plt.show()
```


Model Performance Metrics using LVF
Score
0.30
0.25
0.20
0.15
0.10
0.05
0.00
R² Score
MSE

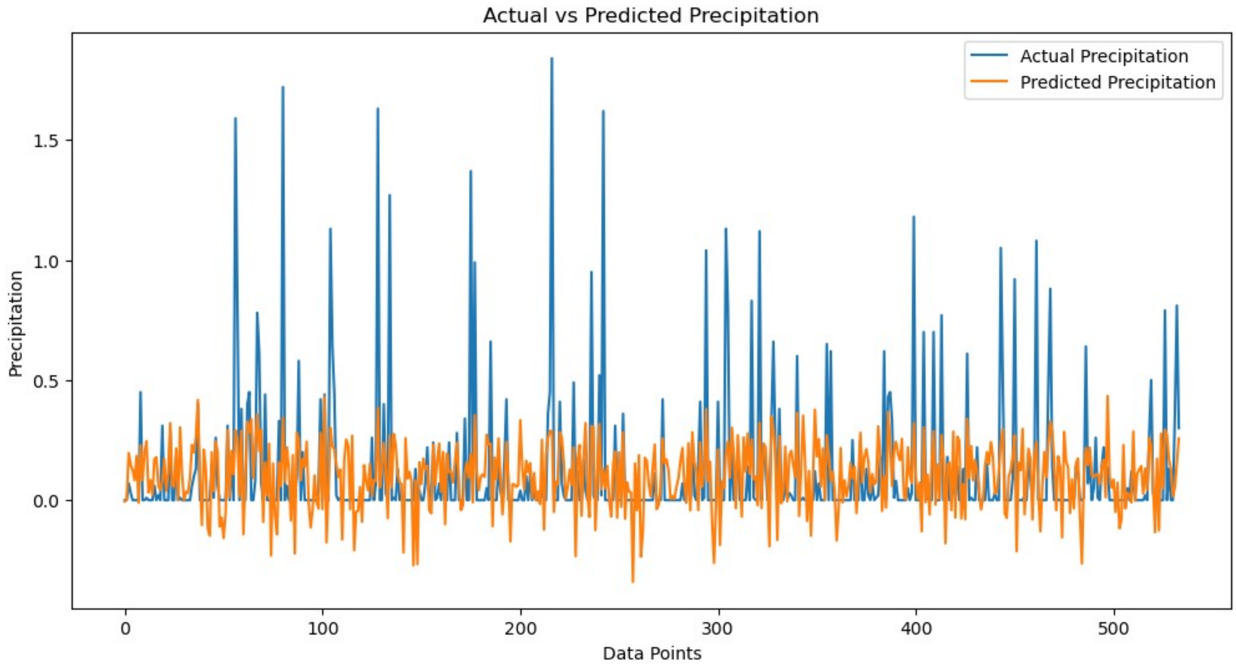



```
# Data for plotting
methods = ['Low Variance Filter (LVF)', 'High Correlation Filter
(HCF)']
r2_scores = [r2LVF, r2HCF]
mse_values = [mseLVF, msehcf]

import matplotlib.pyplot as plt
import numpy as np

# Create a figure with subplots
fig, ax = plt.subplots(1, 2, figsize=(12, 5))

# Bar plot for R² Scores
ax[0].bar(methods, r2_scores, color=['skyblue', 'salmon'])
ax[0].set_ylim(0, 1)  # R² Score ranges from 0 to 1
ax[0].set_title('R² Score Comparison')
ax[0].set_ylabel('R² Score')
for i, v in enumerate(r2_scores):
    ax[0].text(i, v + 0.01, f'{v:.2f}', ha='center', fontsize=12)

# Bar plot for Mean Squared Error (MSE)
```

```
ax[1].bar(methods, mse_values, color=['lightgreen', 'orange'])
ax[1].set_title('Mean Squared Error Comparison')
ax[1].set_ylabel('Mean Squared Error')
# Show the plots
plt.tight_layout()
plt.show()
```




```
plt.figure(figsize=(12, 6))
plt.plot(y_test.values, label="Actual Precipitation")
plt.plot(y_pred, label="Predicted Precipitation")
plt.legend()
plt.title("Actual vs Predicted Precipitation")
plt.xlabel("Data Points")
plt.ylabel("Precipitation")
plt.show()
```




```python
import seaborn as sns
plt.figure(figsize=(12, 8))
corr_matrix = X_selected.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap of Selected Features")
plt.show()
```

Correlation Heatmap of Selected Features