In [1]:
```
import pandas as pd
dataset=pd.read_csv("pre_crop_yield.csv")
dataset
```

Out[1]:

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest	Yield_tons_per_hectare
0	West	Sandy	Cotton	897.077239	27.676966	False	True	Cloudy	122	6.555816
1	South	Clay	Rice	992.673282	18.026142	True	True	Rainy	140	8.527341
2	North	Loam	Barley	147.998025	29.794042	False	False	Sunny	106	1.127443
3	North	Sandy	Soybean	986.866331	16.644190	False	True	Rainy	146	6.517573
4	South	Silt	Wheat	730.379174	31.620687	True	True	Cloudy	110	7.248251
...	...	...	...	...	...	...	...	...	...	...
999984	West	Silt	Rice	302.805345	27.987428	False	False	Sunny	76	1.347586
999985	South	Chalky	Barley	932.991383	39.661039	True	False	Rainy	93	7.311594
999986	North	Peaty	Cotton	867.362046	24.370042	True	False	Cloudy	108	5.763182
999987	West	Silt	Wheat	492.812857	33.045505	False	False	Sunny	102	2.070159
999988	West	Sandy	Maize	180.936180	27.298847	True	False	Sunny	76	2.937243

999989 rows × 10 columns

In [2]:
```
dataset.isnull().sum()
```

Out[2]:
```
Region                 0
Soil_Type              0
Crop                   0
Rainfall_mm            0
Temperature_Celsius    0
Fertilizer_Used        0
Irrigation_Used        0
Weather_Condition      0
Days_to_Harvest        0
Yield_tons_per_hectare 0
dtype: int64
```

In [3]:
```
dataset.columns
```

Out[3]:
```
Index(['Region', 'Soil_Type', 'Crop', 'Rainfall_mm', 'Temperature_Celsius',
       'Fertilizer_Used', 'Irrigation_Used', 'Weather_Condition',
       'Days_to_Harvest', 'Yield_tons_per_hectare'],
      dtype='object')
```

In [4]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Assuming 'dataset' is your DataFrame
categorical_columns = ['Crop','Soil_Type','Region','Weather_Condition']

for column in categorical_columns:
    le = LabelEncoder()
    dataset[column] = le.fit_transform(dataset[column])
dataset
```

Out[4]:

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest	Yield_tons_per_hectare
0	3	4	1	897.077239	27.676966	False	True	0	122	6.555816
1	2	1	3	992.673282	18.026142	True	True	1	140	8.527341
2	1	2	0	147.998025	29.794042	False	False	2	106	1.127443
3	1	4	4	986.866331	16.644190	False	True	1	146	6.517573
4	2	5	5	730.379174	31.620687	True	True	0	110	7.248251
...	...	...	...	...	...	...	...	...	...	...
999984	3	5	3	302.805345	27.987428	False	False	2	76	1.347586
999985	2	0	0	932.991383	39.661039	True	False	1	93	7.311594
999986	1	3	1	867.362046	24.370042	True	False	0	108	5.763182
999987	3	5	5	492.812857	33.045505	False	False	2	102	2.070159
999988	3	4	2	180.936180	27.298847	True	False	2	76	2.937243

999989 rows × 10 columns

In [5]:
```
dataset=pd.get_dummies(dataset,drop_first=True)
dataset
```

Out[5]:

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest	Yield_tons_per_hectare
0	3	4	1	897.077239	27.676966	False	True	0	122	6.555816
1	2	1	3	992.673282	18.026142	True	True	1	140	8.527341
2	1	2	0	147.998025	29.794042	False	False	2	106	1.127443
3	1	4	4	986.866331	16.644190	False	True	1	146	6.517573
4	2	5	5	730.379174	31.620687	True	True	0	110	7.248251
...	...	...	...	...	...	...	...	...	...	...
999984	3	5	3	302.805345	27.987428	False	False	2	76	1.347586
999985	2	0	0	932.991383	39.661039	True	False	1	93	7.311594
999986	1	3	1	867.362046	24.370042	True	False	0	108	5.763182
999987	3	5	5	492.812857	33.045505	False	False	2	102	2.070159
999988	3	4	2	180.936180	27.298847	True	False	2	76	2.937243

999989 rows × 10 columns

In [6]:
```
dataset.columns
```

Out[6]:
```
Index(['Region', 'Soil_Type', 'Crop', 'Rainfall_mm', 'Temperature_Celsius',
       'Fertilizer_Used', 'Irrigation_Used', 'Weather_Condition',
       'Days_to_Harvest', 'Yield_tons_per_hectare'],
      dtype='object')
```

In [7]:
```
independent=dataset[['Region', 'Soil_Type', 'Crop', 'Rainfall_mm', 'Temperature_Celsius','Fertilizer_Used', 'Irrigation_Used', 'Weather_Condition','Days_to_Harvest']]
```

In [8]:
```
independent
```

Out[8]:

	Region	Soil_Type	Crop	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Weather_Condition	Days_to_Harvest
0	3	4	1	897.077239	27.676966	False	True	0	122
1	2	1	3	992.673282	18.026142	True	True	1	140
2	1	2	0	147.998025	29.794042	False	False	2	106
3	1	4	4	986.866331	16.644190	False	True	1	146
4	2	5	5	730.379174	31.620687	True	True	0	110
...	...	...	...	...	...	...	...	...	...
999984	3	5	3	302.805345	27.987428	False	False	2	76
999985	2	0	0	932.991383	39.661039	True	False	1	93
999986	1	3	1	867.362046	24.370042	True	False	0	108
999987	3	5	5	492.812857	33.045505	False	False	2	102
999988	3	4	2	180.936180	27.298847	True	False	2	76

999989 rows × 9 columns

In [9]:
```
dependent=dataset[["Yield_tons_per_hectare"]]
dependent
```

Out[9]:

	Yield_tons_per_hectare
0	6.555816
1	8.527341
2	1.127443
3	6.517573
4	7.248251
...	...
999984	1.347586
999985	7.311594
999986	5.763182
999987	2.070159
999988	2.937243

999989 rows × 1 columns

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(independent, dependent, test_size = 1/3, random_state = 0)
```

In [11]:
```
#model creation phase and linearregression library
from sklearn.linear_model import LinearRegression
#linearregression function assign on regressor
regressor=LinearRegression()
#fit is an train dataset model
regressor.fit(X_train,y_train)
#weight linear regression
weight=regressor.coef_
#weight result
weight
```

Out[11]:
```
array([[-6.87801878e-04, -5.01608032e-04, -1.02165235e-04,
         4.79624825e-01,  1.99257292e-02,  1.45975051e+00,
         1.20101846e+00,  1.05120212e-03,  4.86305900e-05]])
```

In [12]:
```
#bias or initial value or minimum value
bias=regressor.intercept_
#bias or initial value result
bias
```

Out[12]:
```
array([0.00063438])
```

In [13]:
```
y_pred=regressor.predict(X_test)
```

In [14]:
```
#R2 value or better model creation and r2 library
from sklearn.metrics import r2_score
r_score=r2_score(y_test , y_pred)
#r2 result
r_score
```

Out[14]:
```
0.9127672311875132
```

Feature Selection

In [15]:
```
import pandas as pd
from sklearn.feature_selection import RFE, SelectKBest, f_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score


dataset1 = pd.read_csv("pre_crop_yield.csv")
df2 = pd.get_dummies(dataset1, drop_first=True)

indep_X = df2.drop('Yield_tons_per_hectare', axis=1)
dep_Y = df2['Yield_tons_per_hectare']

def split_scalar(indep_X, dep_Y):
    X_train, X_test, y_train, y_test = train_test_split(indep_X, dep_Y, test_size=0.25, random_state=0)
    sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
    return X_train, X_test, y_train, y_test

def rfeFeature(indep_X, dep_Y, n_features):
    model = LinearRegression()
    rfe = RFE(estimator=model, n_features_to_select=n_features)
    fit = rfe.fit(indep_X, dep_Y)
    selected_features = indep_X.columns[fit.support_]
    return selected_features

# SelectKBest for feature selection
def select_k_best_features(indep_X, dep_Y, n_features):
    selector = SelectKBest(score_func=f_regression, k=n_features)
    selector.fit(indep_X, dep_Y)
    selected_features = indep_X.columns[selector.get_support()]
    return selected_features


def r2_prediction(regressor, X_test, y_test):
    y_pred = regressor.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    return r2


def Linear(X_train, y_train, X_test, y_test):
    regressor = LinearRegression()
    regressor.fit(X_train, y_train)
    r2 = r2_prediction(regressor, X_test, y_test)
    return r2

def svm_linear(X_train, y_train, X_test, y_test):
    from sklearn.svm import SVR
    regressor = SVR(kernel='linear')
    regressor.fit(X_train, y_train)
    r2 = r2_prediction(regressor, X_test, y_test)
    return r2

def svm_hi(X_train, y_train, X_test, y_test):
    from sklearn.svm import SVR
    regressor = SVR(kernel='rbf')
    regressor.fit(X_train, y_train)
    r2 = r2_prediction(regressor, X_test, y_test)
    return r2

def Decision(X_train, y_train, X_test, y_test):
    from sklearn.tree import DecisionTreeRegressor
    regressor = DecisionTreeRegressor(random_state=0)
    regressor.fit(X_train, y_train)
    r2 = r2_prediction(regressor, X_test, y_test)
    return r2

def random(X_train, y_train, X_test, y_test):
    from sklearn.ensemble import RandomForestRegressor
    regressor = RandomForestRegressor(n_estimators=10, random_state=0)
    regressor.fit(X_train, y_train)
    r2 = r2_prediction(regressor, X_test, y_test)
    return r2

print(" Identify the top 5 features using SelectKBest")

top_features_kbest = select_k_best_features(indep_X, dep_Y, 5)
print("Top 5 features using SelectKBest:", list(top_features_kbest))


X_train_kbest, X_test_kbest, y_train_kbest, y_test_kbest = split_scalar(indep_X[top_features_kbest], dep_Y)

print("-----------------------------------------")
print("Train and evaluate models on the SelectKBest selected features")

kbest_r2_score = Linear(X_train_kbest, y_train_kbest, X_test_kbest, y_test_kbest)
print(f"R2 Score using SelectKBest features linear: {kbest_r2_score:.4f}")
```

```
 Identify the top 5 features using SelectKBest
Top 5 features using SelectKBest: ['Rainfall_mm', 'Temperature_Celsius', 'Fertilizer_Used', 'Irrigation_Used', 'Days_to_Harvest']
-----------------------------------------
Train and evaluate models on the SelectKBest selected features
R2 Score using SelectKBest features linear: 0.9128
```

In [16]:
```
top_5_dataset = df2[list(top_features_kbest) + ['Yield_tons_per_hectare']]

# Save to CSV
top_5_dataset.to_csv("crop_yield_data.csv", index=False)
```

In [17]:
```
top_5_dataset
```

Out[17]:

	Rainfall_mm	Temperature_Celsius	Fertilizer_Used	Irrigation_Used	Days_to_Harvest	Yield_tons_per_hectare
0	897.077239	27.676966	False	True	122	6.555816
1	992.673282	18.026142	True	True	140	8.527341
2	147.998025	29.794042	False	False	106	1.127443
3	986.866331	16.644190	False	True	146	6.517573
4	730.379174	31.620687	True	True	110	7.248251
...	...	...	...	...	...	...
999984	302.805345	27.987428	False	False	76	1.347586
999985	932.991383	39.661039	True	False	93	7.311594
999986	867.362046	24.370042	True	False	108	5.763182
999987	492.812857	33.045505	False	False	102	2.070159
999988	180.936180	27.298847	True	False	76	2.937243

999989 rows × 6 columns

In [ ]: