In [ ]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import LabelEncoder
```

The crop dataset is read after that the location column is label encoded for the model

In [ ]:
```
df = pd.read_csv('cropStats.csv')
df = df.iloc[:, 1:-1]  # Selects all rows, and columns from index 1 to the second-t


label_encoder = LabelEncoder()
df['Location_LabelEncoded'] = label_encoder.fit_transform(df['Location'])

# Dropping the original 'Location' column
df = df.drop('Location', axis=1)

df.insert(loc=0, column='Location_LabelEncoded', value=df.pop('Location_LabelEncode
df
```

Out[ ]:

	Location_LabelEncoded	1900	1901	1902	1903	1904	
0	0	1064308.40	1044074.40	1084542.40	1088589.20	1052168.00	10
1	1	4451.48	4856.16	4046.80	4856.16	4046.80	
2	2	922670.40	886249.20	906483.20	870062.00	837687.60	8
3	3	25090.16	25494.84	25899.52	25090.16	23876.12	
4	4	38039.92	44110.12	53013.08	57869.24	66367.52	
5	5	78912.60	78103.24	77698.56	80126.64	78912.60	
6	6	232691.00	230667.60	238761.20	240784.60	240784.60	2
7	7	1444707.60	1388052.40	1388052.40	1363771.60	1339490.80	14
8	8	4046.80	4046.80	3642.12	4046.80	3237.44	
9	9	4232952.80	4330076.00	4390778.00	4269374.00	4249140.00	42
10	10	2033517.00	2043634.00	2114453.00	2084102.00	2205506.00	20
11	11	3682588.00	3828272.80	3868740.80	3403358.80	3864694.00	38
12	12	3023769.00	2766392.50	2889415.20	2714188.80	2714188.80	28
13	13	1363771.60	1375912.00	1416380.00	1386029.00	1497316.00	14
14	14	526084.00	509896.80	505850.00	489662.80	473475.60	4
15	15	270326.24	271135.60	273159.00	267088.80	269112.20	2
16	16	635347.60	647488.00	655581.60	655581.60	647488.00	6
17	17	598926.40	651534.80	728424.00	724377.20	789126.00	7
18	18	849828.00	857921.60	870062.00	882202.40	829594.00	8
19	19	3116036.00	3055334.00	3055334.00	2852994.00	2691122.00	28
20	20	1214.04	1618.72	2023.40	2023.40	2428.08	
21	21	2974398.00	2903579.00	2974398.00	2863111.00	2944047.00	29
22	22	127878.88	120189.96	125046.12	114929.12	114929.12	1
23	23	18210.60	19829.32	21448.04	23876.12	22662.08	
24	24	341954.60	319697.20	327790.80	307556.80	311603.60	3
25	25	1092636.00	1040027.60	1080495.60	1027887.20	1027887.20	10
26	26	35611.84	55845.84	50989.68	55036.48	60702.00	
27	27	1618720.00	1568135.00	1618720.00	1578252.00	1608603.00	15
28	28	1064308.40	1116916.80	1278788.80	1250461.20	1456848.00	15
29	29	9712.32	10926.36	11331.04	11735.72	11735.72	

	Location_LabelEncoded	1900	1901	1902	1903	1904	
30	30	635347.60	629277.40	635347.60	617137.00	598926.40	5
31	31	704143.20	667722.00	692002.80	663675.20	639394.40	6
32	32	517990.40	586786.00	671768.80	687956.00	708190.00	7
33	33	1311163.20	1343537.60	1428520.40	1392099.20	1384005.60	13
34	34	1954604.40	1938417.20	1974838.40	1974838.40	1954604.40	20
35	35	4451.48	4451.48	4451.48	4451.48	4046.80	
36	36	768892.00	768892.00	776985.60	748658.00	748658.00	7
37	37	6070.20	6474.88	6879.56	7284.24	7688.92	
38	38	297439.80	291369.60	295416.40	279229.20	279229.20	2
39	39	637371.00	659628.40	679862.40	667722.00	659628.40	6
40	40	1618.72	2023.40	2428.08	2428.08	2428.08	

41 rows × 119 columns

The data set is split into features X and Y variables, 2017 being the label = "Y" and the rest which are the features = "X"

```
In [ ]: X = df.drop('2017', axis=1)
        y = df['2017']
```

The data is then split into a training set and testing set, as you can see from the shape test and training sets are in their right proportions

```
In [ ]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

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

```
(32, 118) (9, 118) (32,) (9,)
```

An instance of the the regression model is created and applied to the training data

```
In [ ]: reg = LinearRegression().fit(X_train, y_train)
```

The model is now evaluated on test data, the values for gthe test set are predicted, the Mean Squared error is calculated and the r2 score is calculated.

```
In [ ]: # Imports the metrics from scikit-learn
        from sklearn.metrics import mean_squared_error, r2_score

        # Predicts the target values for the test set
        y_pred = reg.predict(X_test)
```

```
# Calculates the mean squared error
mse = mean_squared_error(y_test, y_pred)

# Calculates the R^2 score
r2 = r2_score(y_test, y_pred)

# Prints the results
print('Mean squared error:', mse)
print('R^2 score:', r2)
```

```
Mean squared error: 2452486252.694315
R^2 score: 0.9884000065100695
```

Another way of calculating the r2 score, this is done by taking the rest data and target values as arguments to the pretrained model

```
In [ ]: # Gets the R^2 score for the test set
        r2 = reg.score(X_test, y_test)

        # Prints the result
        print('R^2 score:', r2)
```

```
R^2 score: 0.9884000065100695
```

Both approaches are meant to yield the same R^2 score when used with the same test set and model, and their objective is to assess how well the model predicts the given test data. Since both approaches use the same inputs to conduct the calculations, the R^2 score should be the same as long as the same data (X_test and y_test) and model (reg) are used.

```
In [ ]: import pickle
        # saves the model to disk
        filename = 'SKLCropModel.pkl'
        pickle.dump(reg, open('saved_models/sci_kit_crop_model.pkl', 'wb'))
```