Machine Learning in Agriculture

Problem definition

You need to determine the outcome of the harvest season, i.e. whether the crop would be healthy (alive), damaged by pesticides or damaged by other reasons.

Variable	Definition
ID	Unique ID
Estimated_Insects Count	Estimated Insects Count per square meter
Crop_Type	Category of crop 0 and 1
Soil Type	Category of soil 0 and 1
Pesticide_Use_Category	Types of pesticide uses 1-Never 2-Previouslyly used
Number_Doses_Week	Number of Doses Week
Number_Weeks_Used	Number of Weeks Used
Number_Weeks_Quit	Number Weeks Quit
Season	Season Category 1,2 and 3
Crop_Damage	Crop Damage category 0-alive,1-damaged due to other cause, 2-damaged due to pesticide

1. Data Analysis

1.1. Import library and dataset

First step is to import library such as pandas, numpy, seaborn and matplotlib. We need to create CSV file for train data and test data.

Import train dataset as per the below steps (Test dataset can be imported later).

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

```
d=pd.read_csv('trainagri.csv')
d
```

1.2. d.shape: Represent the number of rows and column in the dataset. In our dataset we have 4599 rows and 10 columns.

1.3. d.column: Represent the names of the columns present. We have 10 columns as ID, Estimated_Insects_Count, Crop_Type, Soil Type, Pesticide_Use_Category Number_Doses_Week, Number_Weeks_Used, Number_Weeks_Quit, Season, Crop_Damage.

1.4. d.info: Represent the data type of each column.

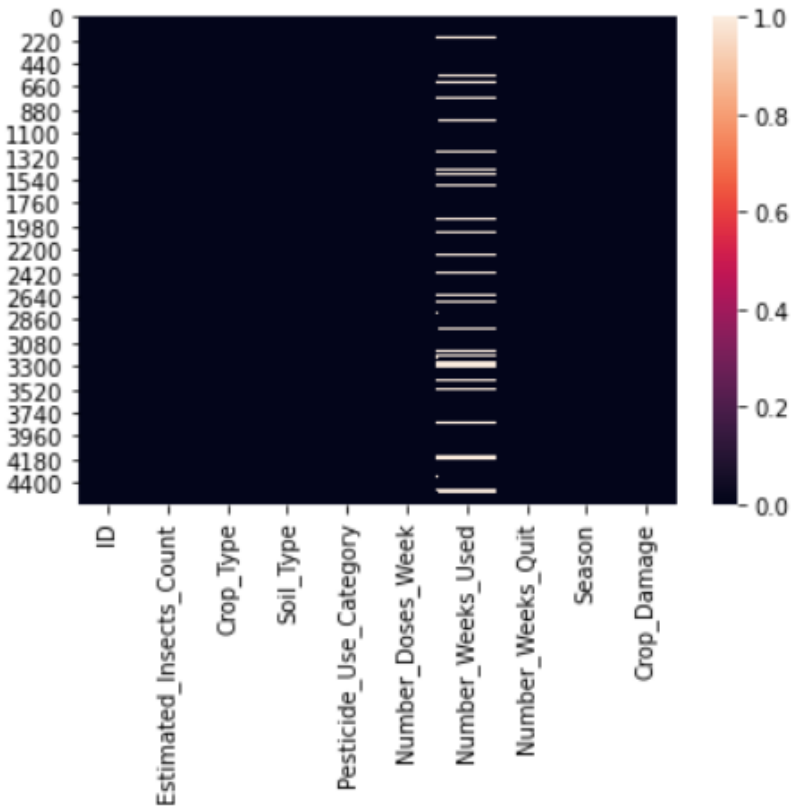
- Object: ID
- Float: Number_Weeks_Used
- Int:
  - Estimated_Insects_Count
  - Crop_Type, Soil_Type
  - Pesticide_Use_Category
  - Number_Doses_Week
  - Number_Weeks_Quit
  - Season
  - Crop_Damage

2. Exploratory Data Analysis

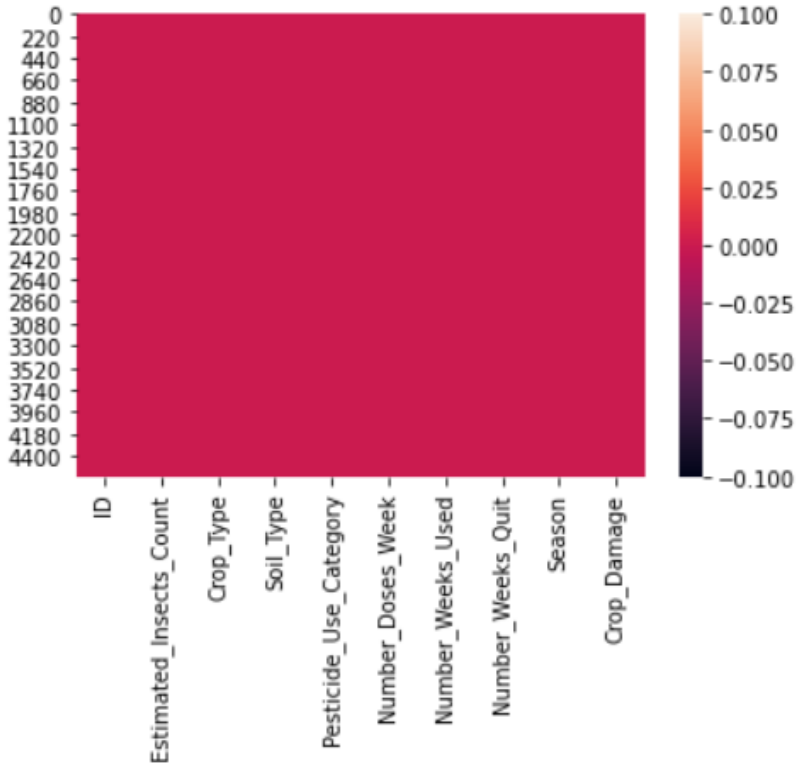
2.1 Check null values

For the given dataset null value present needs to be checked and sum of the null value in each column will be found. Less number of values missed, we can drop those rows. More number of values missed then we can replace numeric value with mean and characters with mode.

sns.heatmap (df.is null()) is used for visual representation of null values.



In our dataset, we have null value for column" Number_Weeks_Used", and we have 442 nan value. Since it is huge when compared to the whole dataset, we replace the NAN value with mean of that column hence we use simple impute method. Later again we check the null value this time there is no null value present.



2.2 Statistics summary

This can be found by d.describe(), where we get to find the mean, median, count, standard deviation and other statistics information.

	Estimated_Insects_Count	Crop_Type	Soil_Type	Pesticide_Use_Category	Number_Doses_Week	Number_Weeks_Used	Number_Weeks_Quit	Season	Crop_Damage
count	4599.000000	4599.000000	4599.000000	4599.000000	4599.000000	4599.000000	4599.000000	4599.000000	4599.000000
mean	1363.000435	0.224831	0.476625	2.283540	26.477495	28.891027	9.205479	1.894325	0.194390
std	814.439120	0.417517	0.499508	0.471978	15.524647	11.965785	9.713253	0.694952	0.454597
min	150.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000
25%	731.000000	0.000000	0.000000	2.000000	15.000000	20.000000	0.000000	1.000000	0.000000
50%	1212.000000	0.000000	0.000000	2.000000	20.000000	28.891027	7.000000	2.000000	0.000000
75%	1786.000000	0.000000	1.000000	3.000000	40.000000	36.000000	16.000000	2.000000	0.000000
max	4097.000000	1.000000	1.000000	3.000000	95.000000	66.000000	47.000000	3.000000	2.000000

From the above table we can infer

- Mean is greater than median for estimated Insects Count: Data might be right skewed
- Difference between 75 and max is great for Estimated Insects Count, Number Doses Week, Number Weeks Quit, Number Weeks Used: Outliers may be there
- Estimated Insects Count min is 150 and max 4097
- Number of weeks used min 0 and max 66
- Number of weeks quit min 0 and max 47

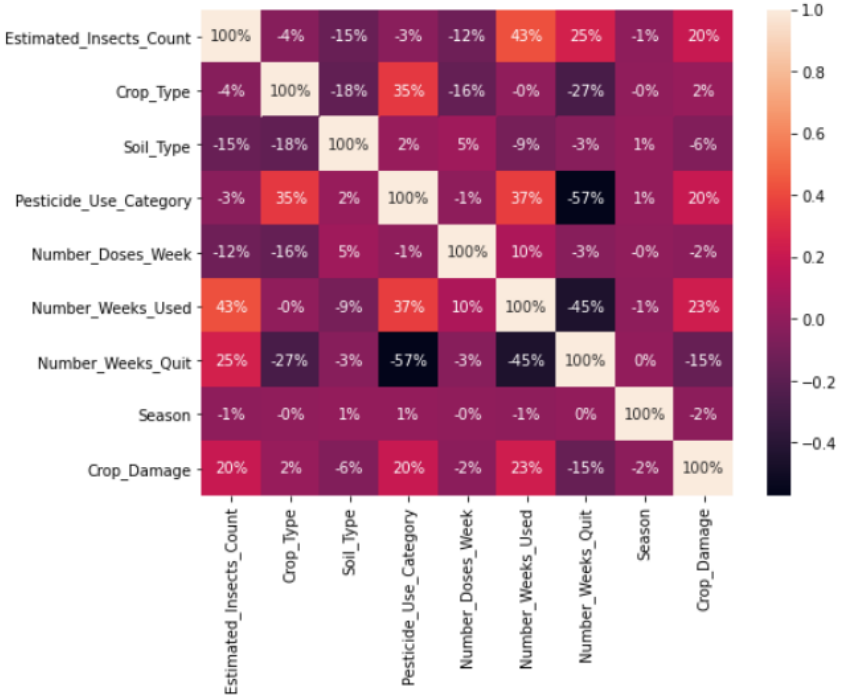
2.3 Correlation

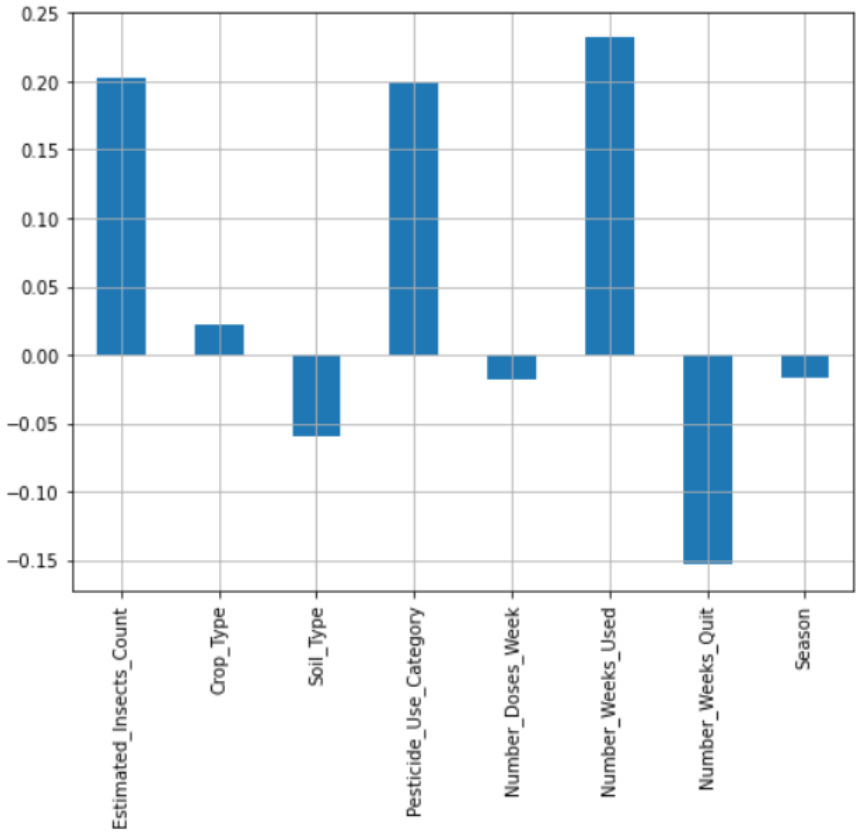
Correlation is used to find how each variable is related with the target variable. It will positively, negatively or neutral correlated. Lighter shade towards the top are positively correlated and darker shared towards the end are negatively correlated.

In our dataset we take Crop damage as target variable. ID column can be dropped as it is not integer type. Correlation and outliers can be done only for integer data. Below charts is used to explain the correlation.

```
: #Correlation
  dfcor=d.corr()
  plt.figure(figsize=(8,6))
  sns.heatmap(dfcor,annot=True,fmt='.0%')
```

```
: plt.figure(figsize=(8,6))
  d.drop('Crop_Damage',axis=1).corrwith(d['Crop_Damage']).plot(kind='bar',grid=True)
```





From the above two correlation chart, we can understand

- Highly positively correlated: Estimated Insects Count (20%), Pesticide Use Category (20%), Number Week used (23%)
- Highly negatively correlated Number Weeks Quit (-15%)
- Negatively correlated: Season (-2%), Soil Type (-6%), Number Doses Week (-2%)

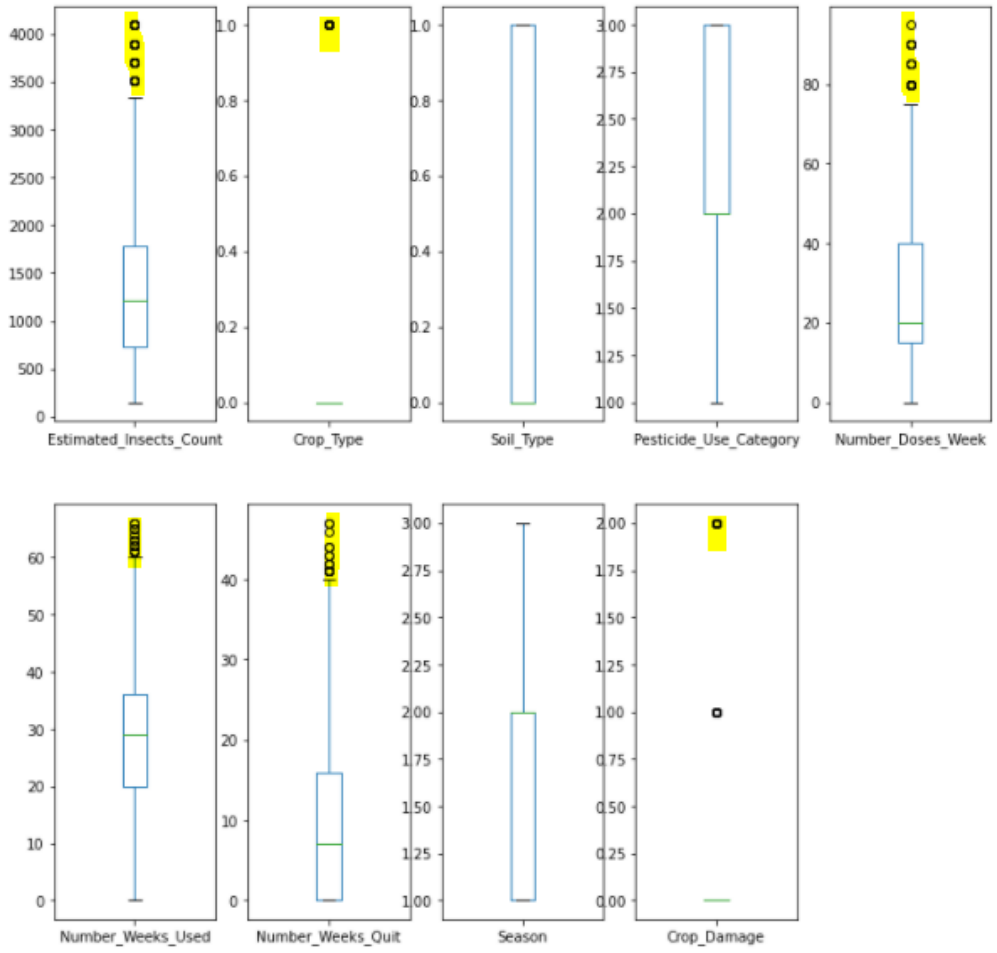
2.4 Outliers

Outliers are those extreme values which lies beyond 3 and -3. Visualization of outliers can be seen by boxplot where point's lies either above the max or lower the min. Outliers will be removed by Z score method.

Applying the same in our dataset, we identified 229 rows as outliers and removed (highlighted in yellow). Given dataset have 4599 rows and 9 columns, after removing outliers we have 4370 rows and 9 columns.

```
#Outliers
d.plot(kind='box',subplots=True,layout=(2,5),figsize=(12,12))
```

```
Estimated_Insects_Count      AxesSubplot(0.125,0.536818;0.133621x0.343182)
Crop_Type                  AxesSubplot(0.285345,0.536818;0.133621x0.343182)
Soil_Type                   AxesSubplot(0.44569,0.536818;0.133621x0.343182)
Pesticide_Use_Category     AxesSubplot(0.606034,0.536818;0.133621x0.343182)
Number_Doses_Week          AxesSubplot(0.766379,0.536818;0.133621x0.343182)
Number_Weeks_Used                AxesSubplot(0.125,0.125;0.133621x0.343182)
Number_Weeks_Quit              AxesSubplot(0.285345,0.125;0.133621x0.343182)
Season                          AxesSubplot(0.44569,0.125;0.133621x0.343182)
Crop_Damage                    AxesSubplot(0.606034,0.125;0.133621x0.343182)
dtype: object
```

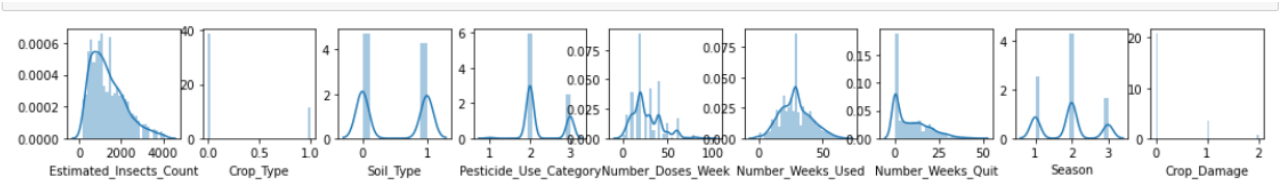


2.5. Skewness

Skewness occurs because data is not of symmetric distribution it would be either towards right as right skewed or towards left as left skewed. We use power transform, boxcox and log method to treat the dataset for highly skewed data.

- skewness is between -0.5- 0.5, the data is symmetrical.
- If the skewness is between -1 -0.5 and 0.5-1 the data are moderately skewed.
- If the skewness is less than -1 or greater than 1 the data are highly skewed.

As per the below table and chart, Crop damage which is the target variable data is not much skewed and we do need to apply any method to treat it.



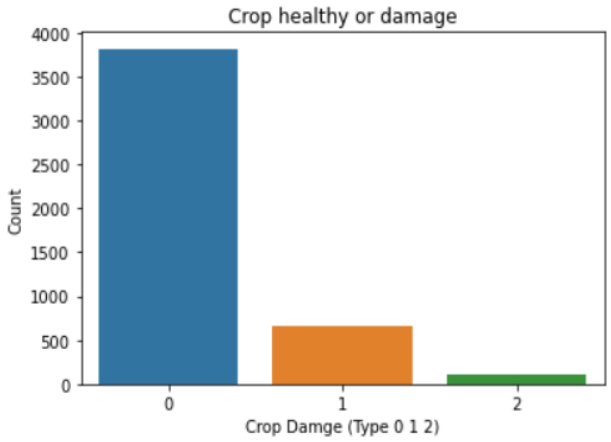
Column	Skewness value
Crop Type	1.318693
Estimated Insects Count	0.911469
Soil Type	0.093631
Pesticide Use Category	0.678586
Number Doses Week	0.945895
Number_Weeks_Used	0.278664
Number_Weeks_Quit	0.919771
Season	0.144841
Crop_Damage	2.306933

3. Univariant and Bivariant analysis

3.1 Univariant analysis

We use Univariant analysis to find the count for crop damage, crop type, soil type, Pesticide Used and season.

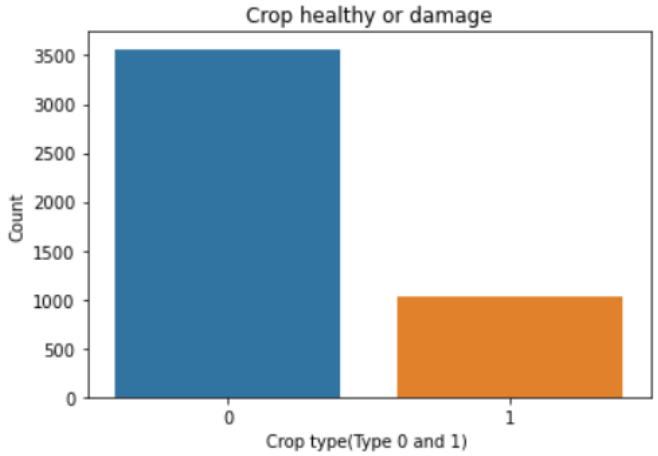
3.1.1 Crop damage




```
0    3820
1     664
2     115
Name: Crop_Damage, dtype: int64
```

From the above barchart we get to know 3820 crops are healthy and 779 crops are damaged due to pesticide and other reasons.

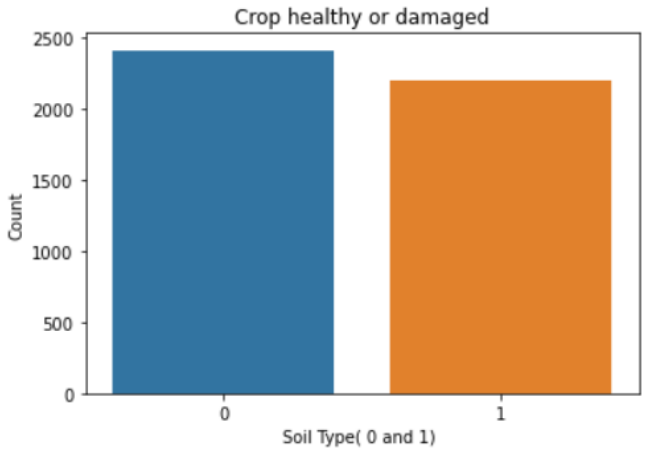
3.1.2 Crop type:




```
0    3565
1    1034
Name: Crop_Type, dtype: int64
```

From the above barchart we get to know there are two types of crop. Type 0 has 3565 crops and Type 1 has 1034 crops.

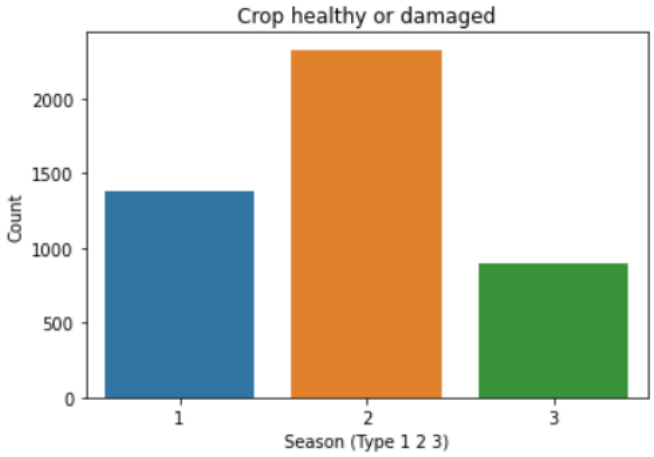
3.1.3 Soil type




```
0    2407
1    2192
Name: Soil_Type, dtype: int64
```

From the above barchart we get to know there are two types of soil. Type 0 has 2407 crops and Type 1 has 2192 crops. Both has almost similar value of crop.

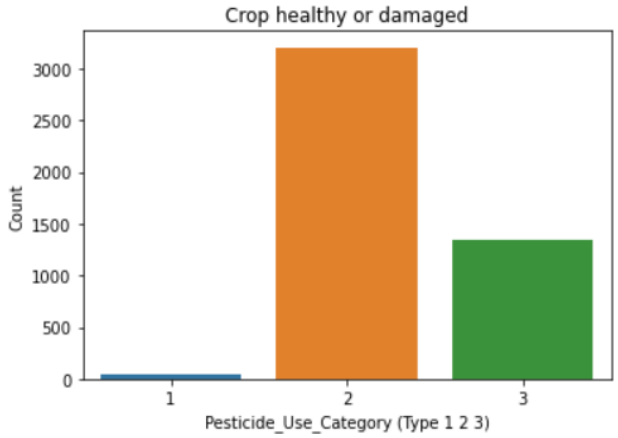
3.1.4 Season




```
2    2327
1    1379
3     893
Name: Season, dtype: int64
```

From the above barchart we get to know there are three seasons. Season 1 has 1379 crop, season 2 has 2327 crop and season 3 has 893. Season 2 has more number when compared to 1 and 3.

3.1.5 Pesticide used

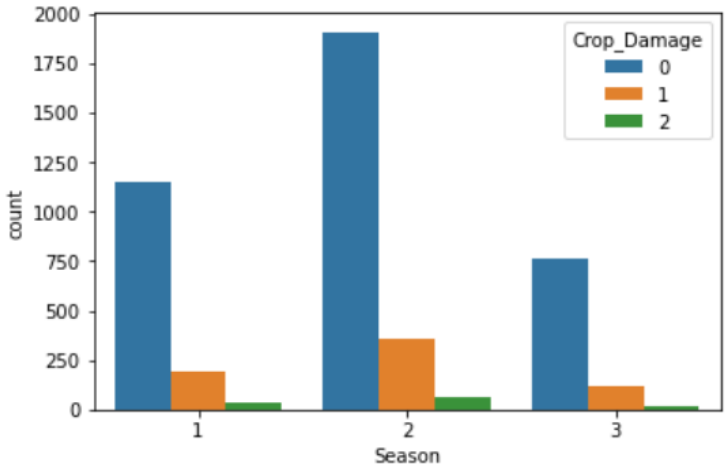



```
2    3205
3    1349
1      45
Name: Pesticide_Use_Category, dtype: int64
```

From the above barchart we get to know there are three types of pesticides. Type 1 has 45, type 2 has 3205 and type 3 has 1349. Type 2 has more number of pesticides when compared to 1 and 3.

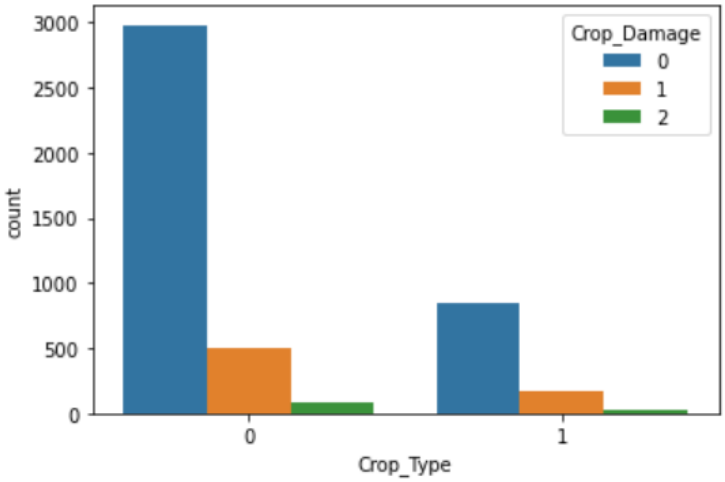
3.2 Bivariant Analysis:

3.2.1 Damaged crop vs season



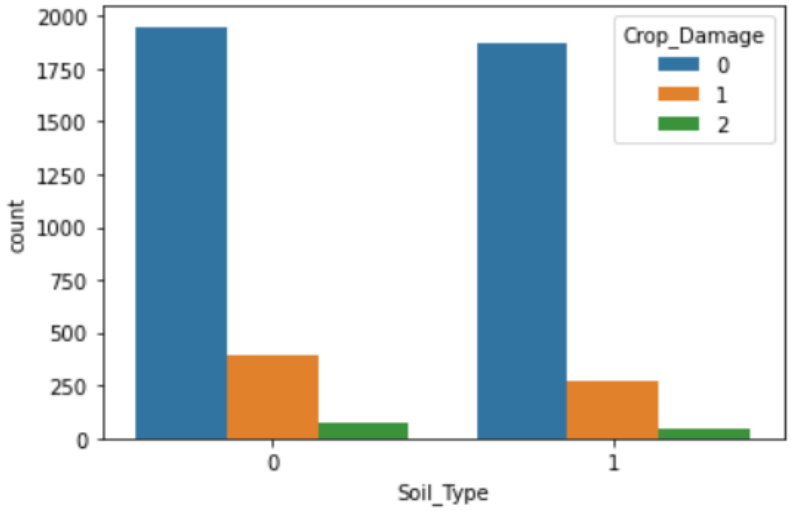
From the above chart we can infer that season 2 has more number of healthy crops and as well as damaged crops which is at 22% when compared to other seasons. The damage percentage of season 1 is 19% and season 3 is 16%. As per analysis season 3 is better.

3.2.2 Damaged crop vs Crop type



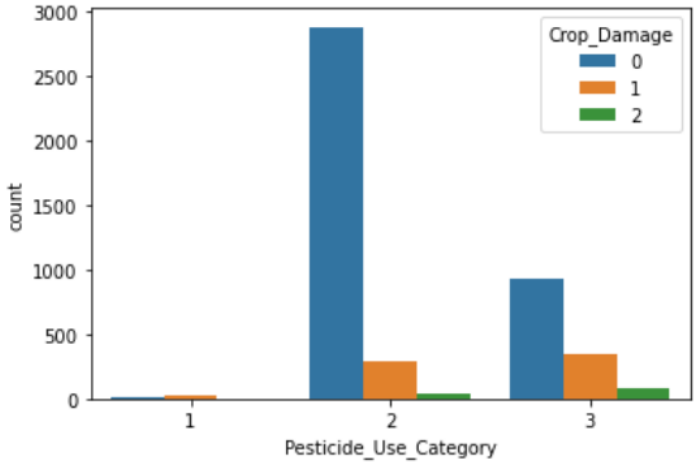
From the above chart we can infer that crop type 0 has more number of alive crops and less number of damaged crop when compared to type 1. The damage percentage of type 0 is 20% whereas as type 1 is 22%.In this case most preffered is crop type is 0.

3.2.3 Damaged crop vs soil type



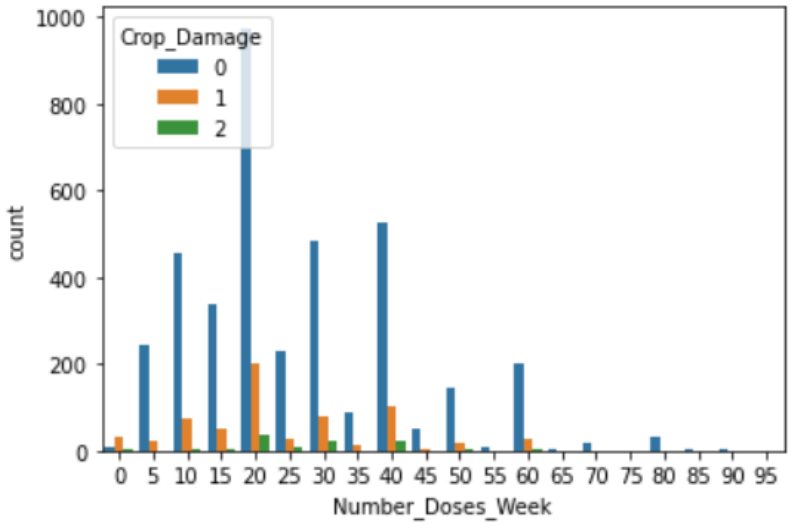
From the above chart we can infer that soil type 1 has less damaged crop and alive crop are almost similar in type 0 and type 1. The damage percentage of type 0 is 24% whereas as type 1 is 11%. In this case most preffered is crop type 1 .

3.2.4 Damaged crop vs Pesticide used



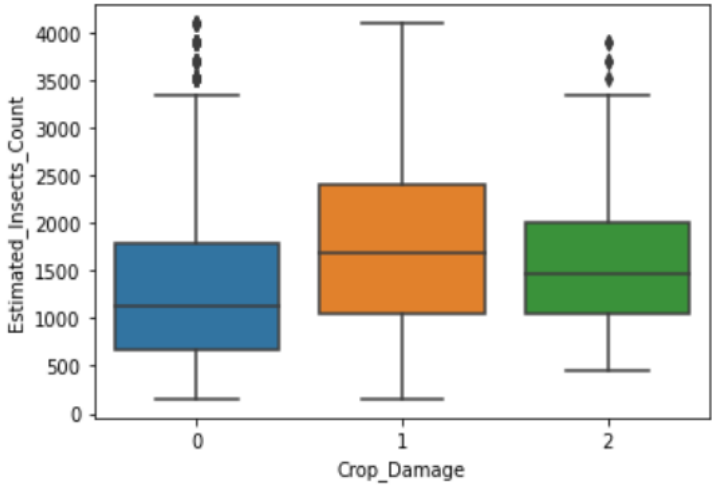
From the above chart we can infer that crop pesticide type 2 has more number of alive crops and less number of damaged plants which is at 11% when to compared to other types of pesticides. The damage percentage of type 1 is 18% whereas as type 3 is 44. In this case most preffered pesticide is type 2.

3.2.5 Damaged crop vs Number of dose week:



From the above chart, Otimal benefit is achieved at 20 weeks dosage. More number of weeks or less number week use of pesticides causes more damages.

3.2.6 Boxplot of Crop damage vs estimated insect count:



From the above boxplot, we can infer the crop healthy or damage is almost at the same line with similar mean min and median. Outliers are present in 0 and 2.

4. Train/Test split

Train/Test split procedure is used to evaluate the performance of various algorithms and used to make prediction on data.

First, we divide the data as input (X) and output (Y). Input consists of all columns except crop damage and output consists of target variable (crop damage). We use standard scaler technique on input data, to make it as normal distribution with mean 0 and standard deviation 1.

On X and Y data we split again with respect to train and test (x_train, x_test, y_train, y_test). Split percentage is of our choice. I have taken train as 70% and test as 30%. Train Dataset is used to train the model and Test data is used to predict the output when input is provided, and result is compared to the expected output.

We have 4370 rows and 9 columns. On splitting data with train and test we have:

- x_train.shape consists of 3059 rows and 8 columns
- x_test.shape consists of 1311 rows and 8 columns
- y_train.shape consists of 3059 rows and 1 columns
- y_test.shape consists of 1311 rows and 1 columns

5. Building model

Our Output (Crop damage) is categorical data which has type 0 as alive, 1 as damaged by pesticides and 2 damaged by others. Hence, I go with classification model.

5.1 Import library

a. From sklearn.metrics- Import accuracy score, confusion matrix, classification report
b. From sklearn.ensemble- Import AdaBoost Classifier, Random Forest Classifier, Gradient Boosting Classifier, Bagging Classifier, Extra Trees Classifier
c. From sklearn-Import Decision Tree Classifier, SVC, GaussianNB, KNeighbors Classifier

5.2 Classification metrics

1. Accuracy score-is calculated as true results among the total number of cases examined.
2. Confusion matrix-The matrix compares the actual target values with those predicted and diagonal will represent the error.
3. Classification report- consists of precision, recall and f1 score**.** Precision is calculated by (TP/TP+FP). Recall is calculated by (TP/TP+FN) and f1-score- is calculated by 2*((P*R)/(P+R)).

   On comparing the results on various algorithms, Best model will be chosen.

5.3. Classification and boosting method

5.3.1 Classification Method

Classification algorithm is applied on the train data. We have used Decision Tree Classifier, GaussianNB, KNN and SVC.

- Decision Tree Classifier: Is a flowchart tree structure where internal node represents attribute, branch represent decision rule and leaf node represents outcome. We start at the tree root and split the data.
- Naïve Bayes: is based on Bayes' Theorem. Bayes Formula (P(H/X)= P(X/H)P(H) / P(X)). It has GaussianNB and MultinomialNB method.
- KNN: stands for knearest neighbor. Distance is calculated by Euclidian distance. Three steps are followed calculate distance, find the closest neighbors, vote for labels.
- SVC: Support vector machine. We perform classification by finding the hyper-plane that differentiates the two classes. SVC has technique called kernel trick (These functions takes low dimensional input space and transform to higher dimensional space). Kernel type Linear, poly and RBF**.**

5.3.2 Boosting Method

Ensemble technique is used on algorithms to improve the accuracy score and get better results. We have used AdaBoost Classifier, RandomForest Classifier, Gradient Boosting Classifier, Extratree Classifier and bagging Classifier.

In Bagging method, multiple models (weak learners) are trained independently from each other in parallel way and all output is combined to get result. Random forest classifier and bagging classifier are the example of bagging method.

In Boosting method Output From one model is boosted and given as input for next model. It is Sequential method. Ada boost and Gradient boost are example of boosting method.

Algorithm	**Accuracy Score**
DecisionTreeClassifier	**77**
GaussianNB	**82**
KNeighborsClassifie	85
SVC	85
AdaBoostClassifier	**86**
RandomForestClassifier	**85**
GradientBoostingClassifier	**85**
ExtraTreesClassifier	**84**
BaggingClassifier	**83**

Based on the result obtained for accuracy score and f1 scale AdaBoost classifier has the highest score.

5.4 Cross validation

Cross validation (CV) is one of the techniques used to test the effectiveness of a machine learning models. K-Fold CV is where a given data set is split into a K number of sections. We have used K value to be 5. Data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds has been used as the testing set.

Our model has an average accuracy of 85.3% with a standard deviation of 9 %. The standard deviation shows us, how precise the estimates are. We will try to increase its performance by using GridsearchCV.

Algorithm	Cross validation Score
DecisionTreeClassifier	**74**
GaussianNB	**78**
KNeighborsClassifie	82
SVC	85
AdaBoostClassifier	**85.5**
RandomForestClassifier	**83**
GradientBoostingClassifier	**85**
ExtraTreesClassifier	**82**
BaggingClassifier	**82**

Based on the result obtained AdaBoost classifier has the highest score.

5.5 GridsearchCV

GridsearchCV is the process of performing hyper parameter tuning in order to determine the optimal values for a given model. We pass in few parameters such as learning rate and n_estimator. Among the values passed in dictionary optimal values is taken. We have obtained best parameters of learning_rate as 0.1 and n_estimators as 700.

```
#Gridsearch for best parameter
from sklearn.model_selection import GridSearchCV
ab=AdaBoostClassifier()
parameters={'n_estimators':[400,500,600,700],'learning_rate':[1,10,0.1]}
clf=GridSearchCV(ab,parameters)
clf.fit(x,y)
print('best parameters',clf.best_params_)
```

```
best parameters {'learning_rate': 0.1, 'n_estimators': 700}
```

5.6 Best model

Based on the result obtained on GridsearchCV, model with those parameters will be saved. We can use the model to predict the output for test dataset. AdaBoost classifier is chosen as best model. We have obtained the below results:

- accuracy score:86.2
- confusion matrix [[1097 33] , [155 26]]
- Precision-82
- Recall-86
- f1 score-82

6. Test dataset

Similar to train dataset, we need to apply the same steps in test dataset.

- Import the test dataset
- Check for null values- Null value present in Number Weeks Used
- Treat it using simple imputer mean method
- ID column can be dropped
- Check for skewness-no skewed data
- check for outliers-Need to remove Outliers
- Original dataset:(1199, 8)
- After removing outliers (1160, 8)

7. Prediction

df_new1 is the new dataset, now this will be passed to the best model saved (Adaboost classifier) and save the result in .csv file.

```
In [109]: print(df2.shape) #dataset
          print(df_new1.shape) #Removed Outliers

          (1199, 8)
          (1160, 8)

In [110]: abc.predict(df_new1)

Out[110]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [111]: df=pd.DataFrame(predm)
          df.to_csv('Agriculture pred.csv')
```