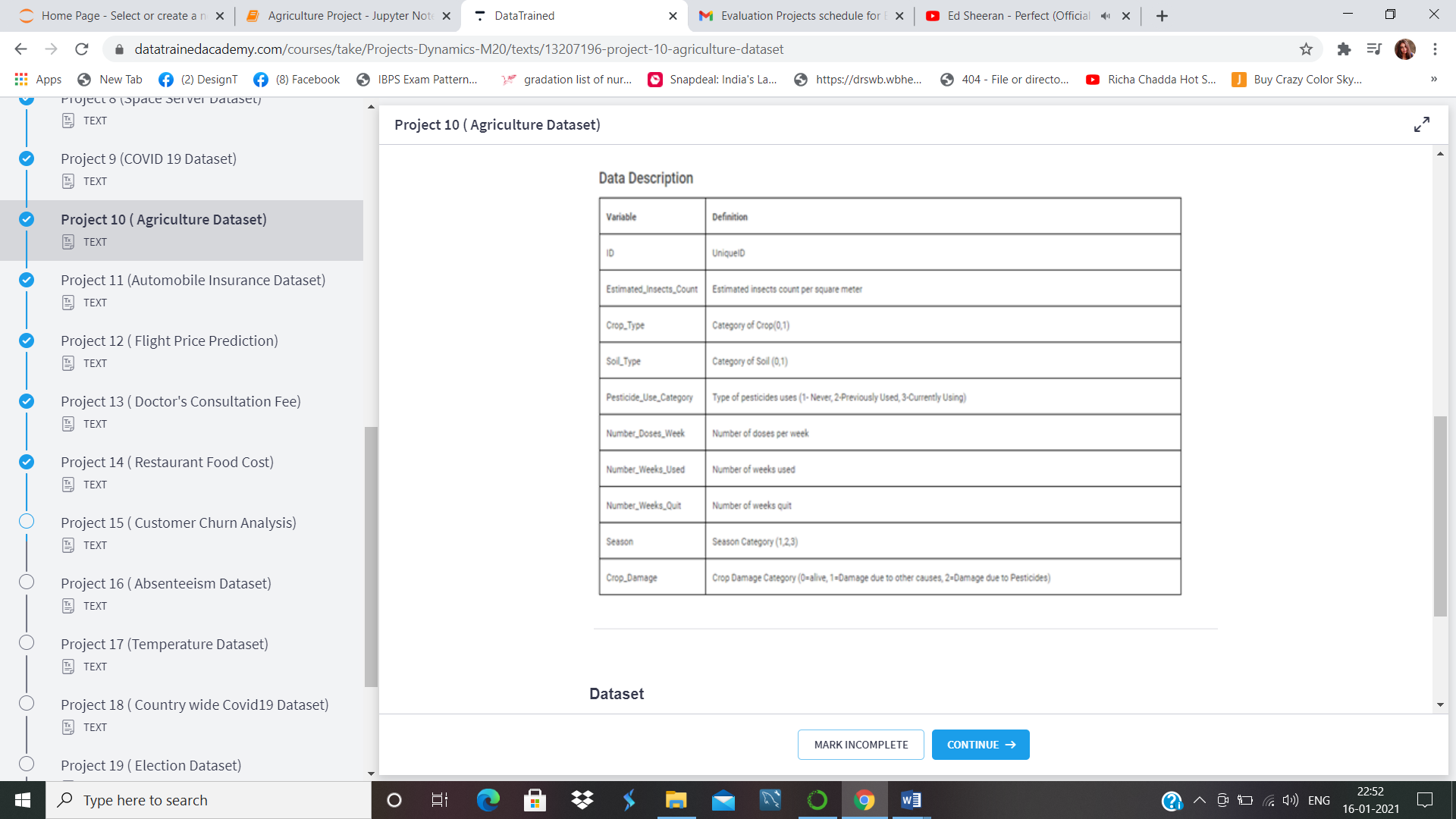
Agriculture Dataset

I am going to make a machine learning model for the Agriculture Dataset. Where we have to determine based on the features given **whether the crop would be healthy (alive), damaged by pesticides or damaged by other reasons.**

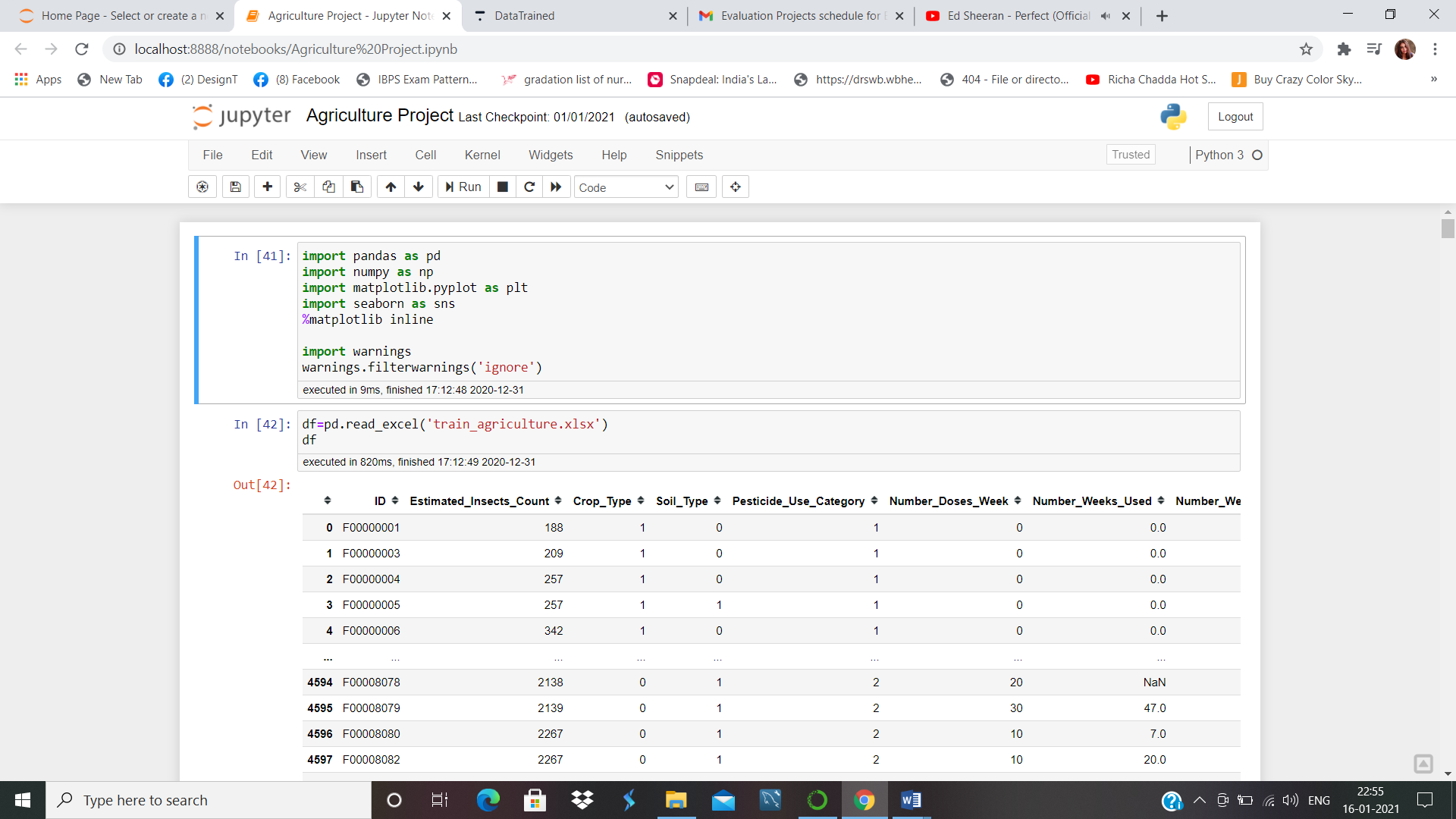
Pesticides are also special, because while they protect the crop with the right dosage. But, if you add more than required, they may spoil the entire harvest. A high level of pesticide can deem the crop dead / unsuitable for consumption among many outcomes. This data is based on crops harvested by various farmers at the end of harvest season.

Data Description:-

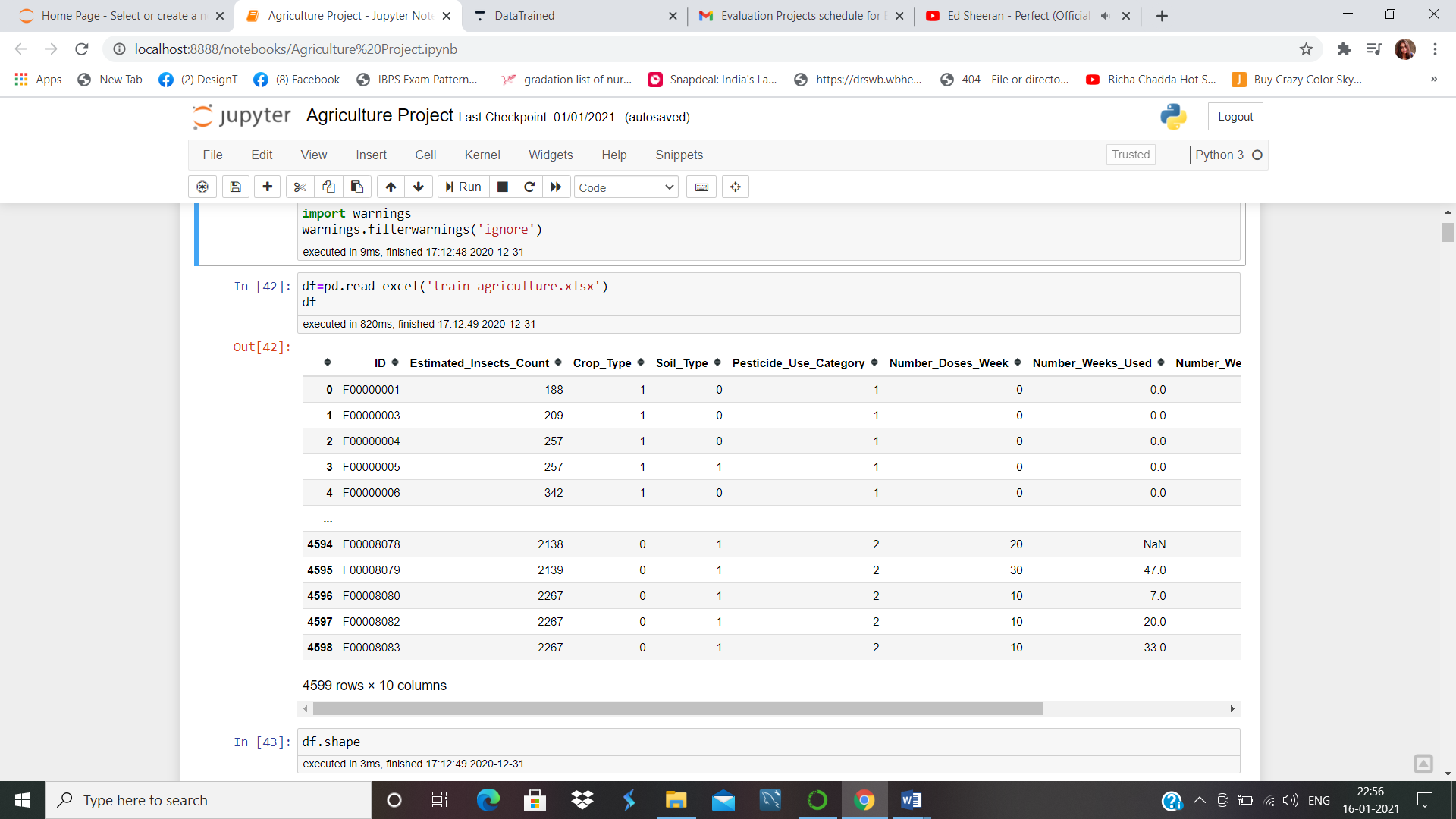


We will import the libraries first.

Importing the library:-



Reading the Dataset:-

Pe 

There are 4599 rows and 10 columns are available among which one column is our target column.

9 features are available to determine whether a crop is damaged or alive.

We will try to find out the insight of the data.

Data Exploration:-

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4599 entries, 0 to 4598

Data columns (total 10 columns):

# Column Non-Null Count Dtype

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

0 ID 4599 non-null object

1 Estimated\_Insects\_Count 4599 non-null int64

2 Crop\_Type 4599 non-null int64

3 Soil\_Type 4599 non-null int64

4 Pesticide\_Use\_Category 4599 non-null int64

5 Number\_Doses\_Week 4599 non-null int64

6 Number\_Weeks\_Used 4157 non-null float64

7 Number\_Weeks\_Quit 4599 non-null int64

8 Season 4599 non-null int64

9 Crop\_Damage 4599 non-null int64

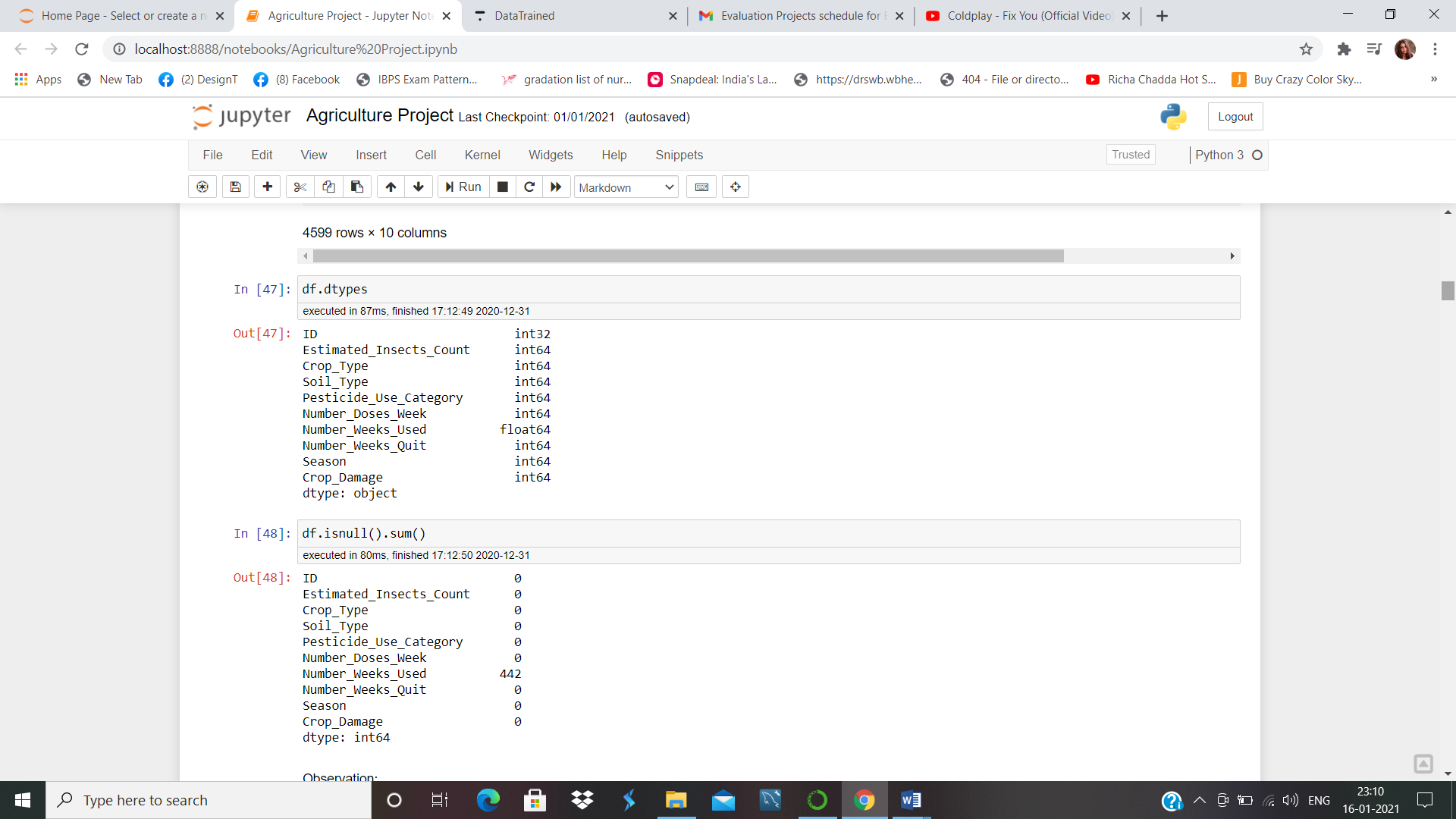
dtypes: float64(1), int64(8), object(1)

There is only one variable ID which is object object type. We will make it integer type by using Label encoder.

**from** **sklearn.preprocessing** **import** LabelEncoder

le=LabelEncoder()

df['ID']=le.fit\_transform(df['ID'])



All the data are now in integer or float type.

We have to see if there is any null value available in the data or not. If there is null value we have to replace it with mean.

df.isnull().sum()

ID 0

Estimated\_Insects\_Count 0

Crop\_Type 0

Soil\_Type 0

Pesticide\_Use\_Category 0

Number\_Doses\_Week 0

Number\_Weeks\_Used 442

Number\_Weeks\_Quit 0

Season 0

Crop\_Damage 0

dtype: int64

There are 442 null values in Number\_Weeks\_Used. We will replace the value with mean and then check the if the null values have got replaced.

df['Number\_Weeks\_Used'].fillna(df['Number\_Weeks\_Used'].mean(),inplace=**True**)

df.isnull().any()

ID False

Estimated\_Insects\_Count False

Crop\_Type False

Soil\_Type False

Pesticide\_Use\_Category False

Number\_Doses\_Week False

Number\_Weeks\_Used False

Number\_Weeks\_Quit False

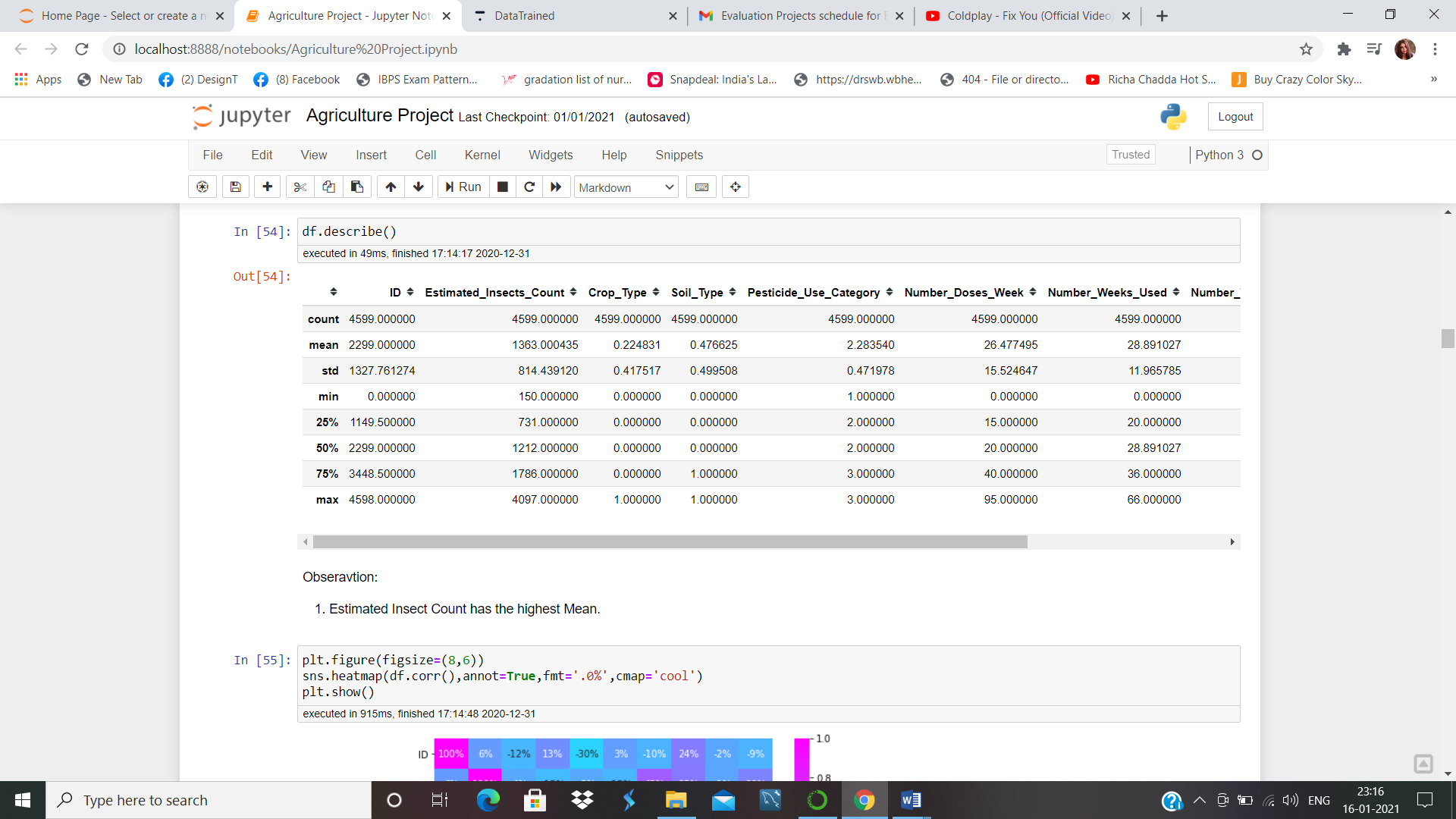
Season False

Crop\_Damage False

dtype: bool

The null values are not present in our data anymore. So we can proceed to EDA.

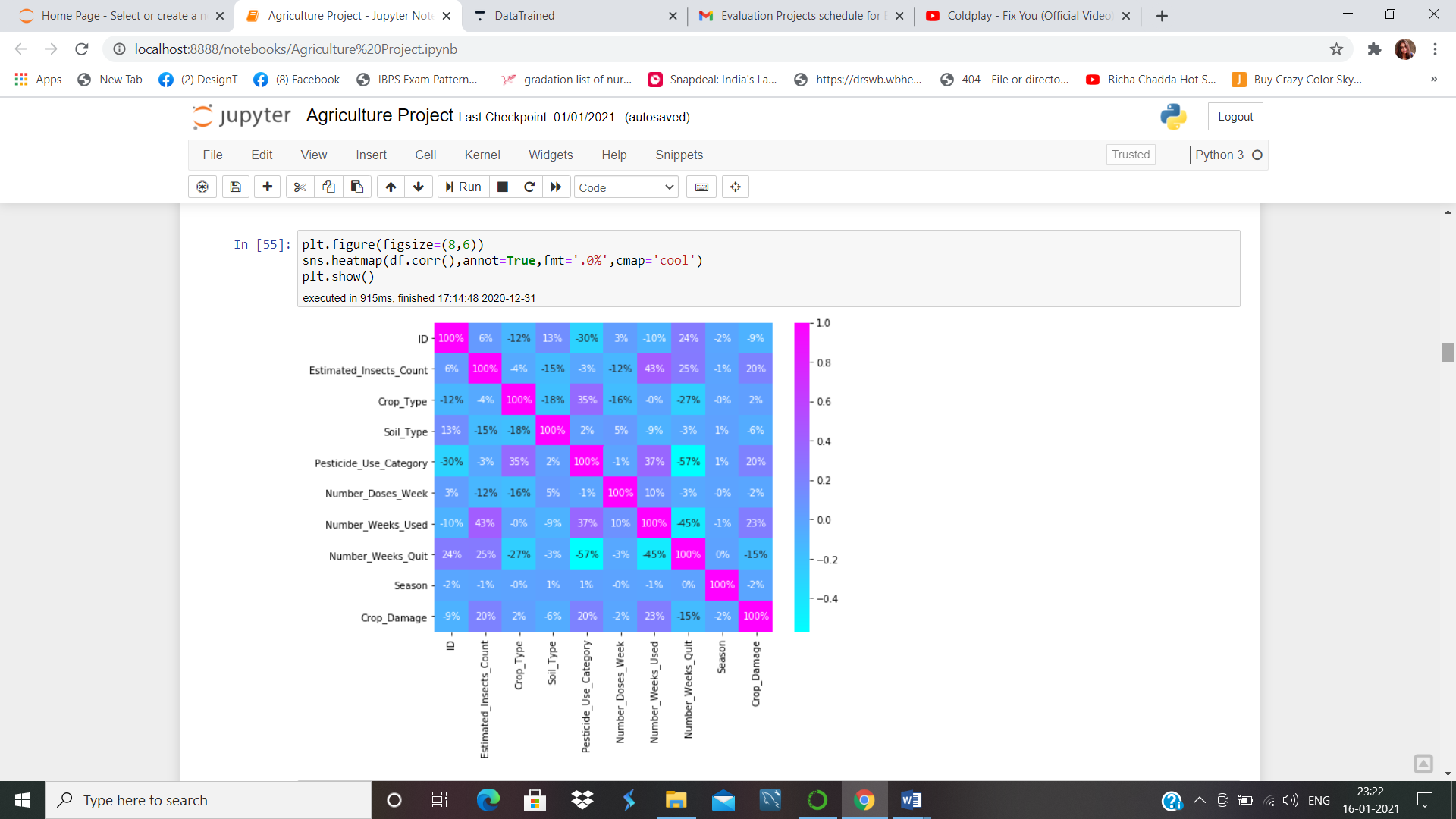
Exploratory Data Analysis:-



Estimated Insect Count has the highest Mean.

Number\_Doses\_Week and Number\_Weeks\_Used have almost same mean value.

Crop\_Type, Soil\_Type, Season, Crop\_Damage are the categorical variable so it has the almost same mean and standard deviation.

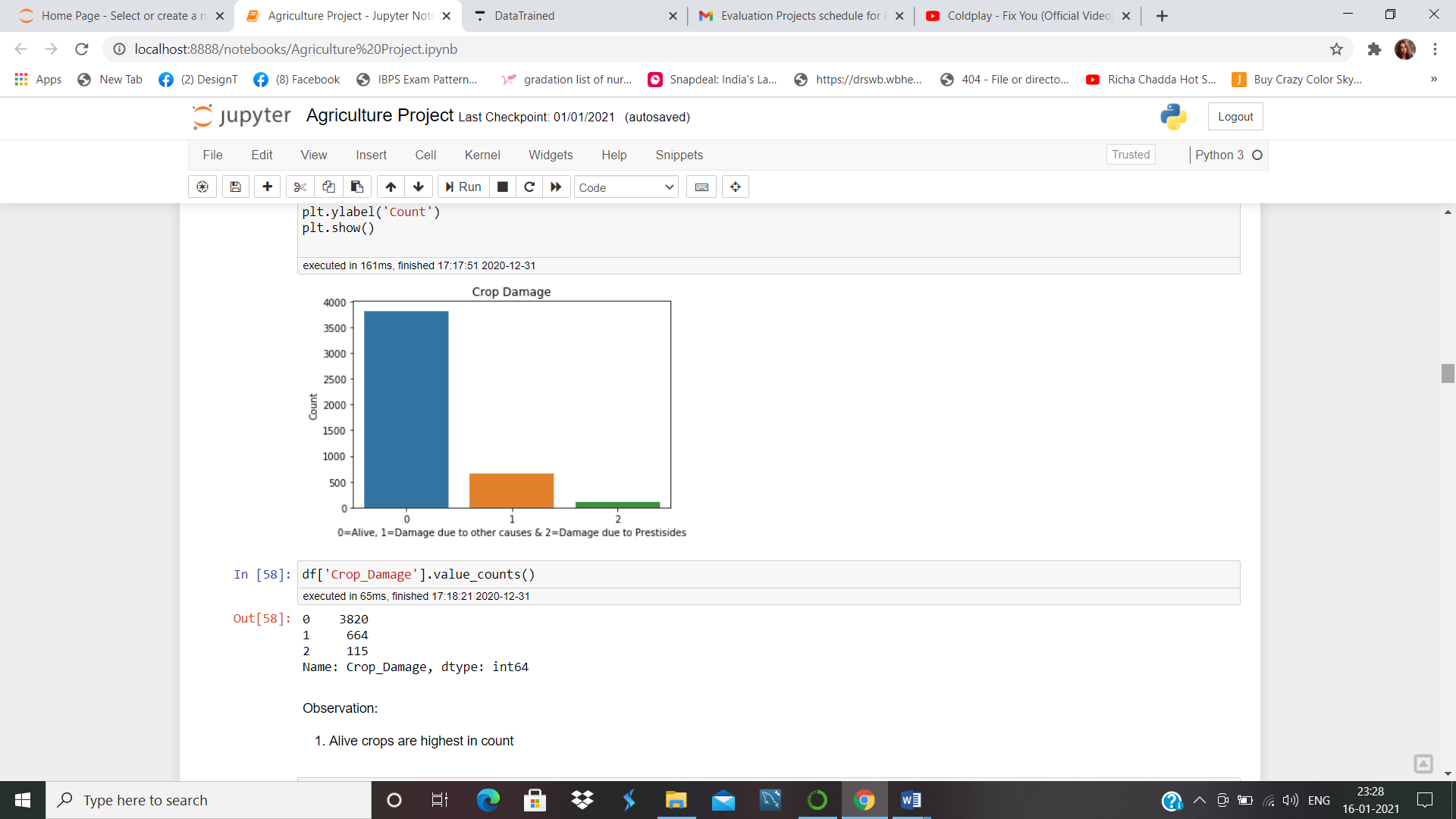


Numbers\_weeks\_used and Estimated\_Insects\_Count have good correlation with 43%.

Crop\_Damage has moderate correlation with Number\_Weeks\_Used, Pesticide\_Use\_Category, Estimated\_Insects\_Counts.

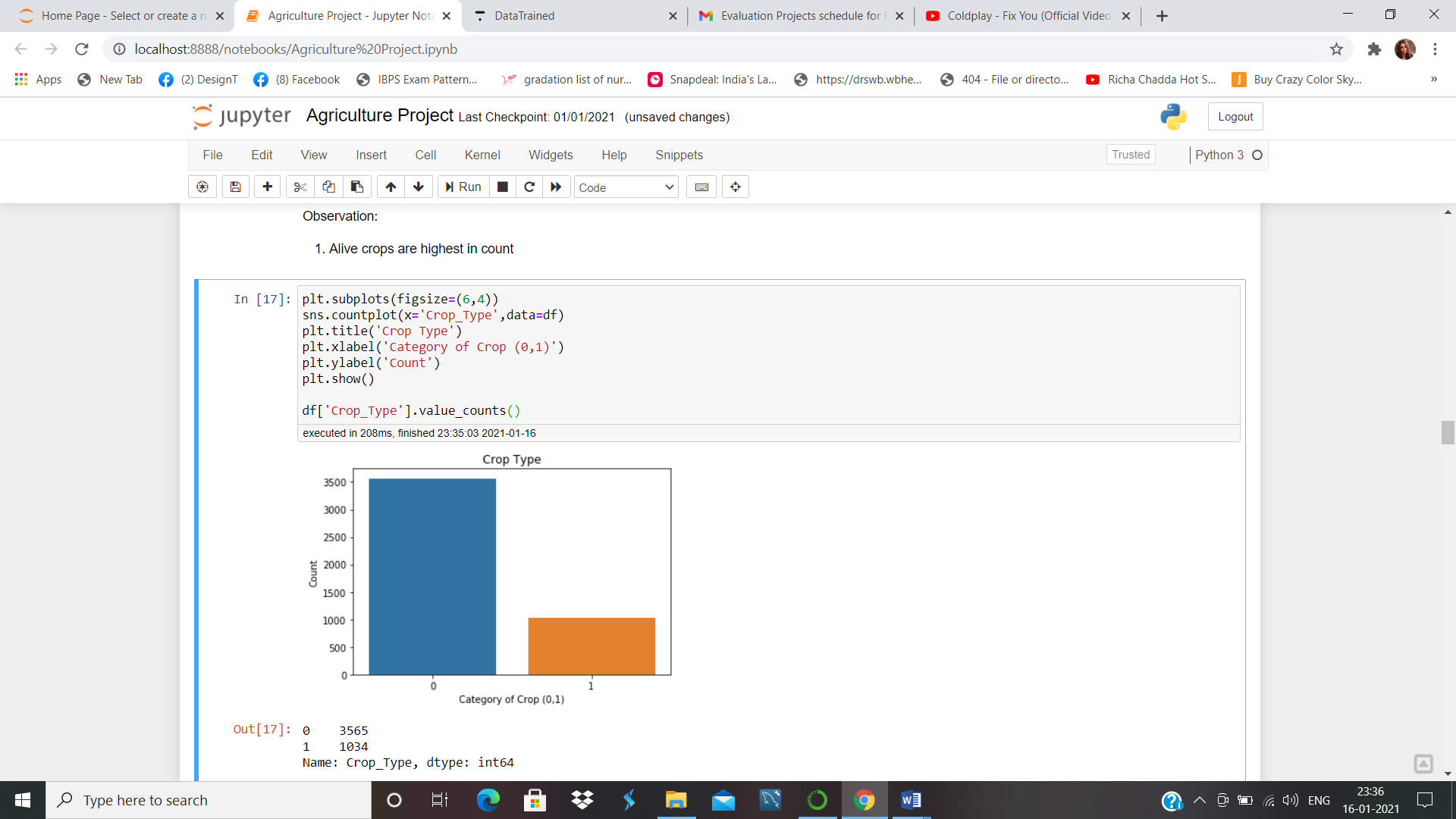
Now we’ll see the countplot of our target variable.



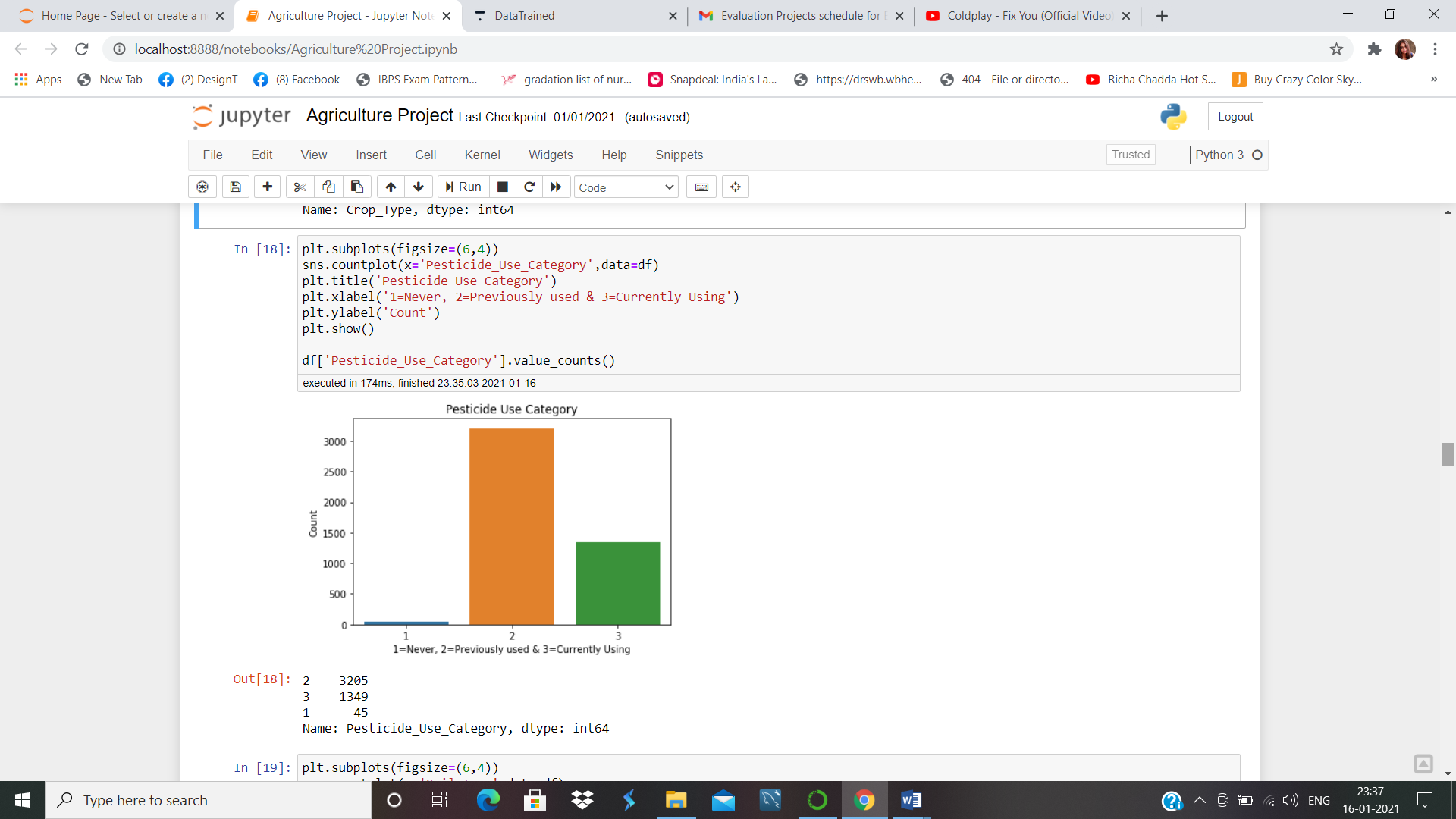


Here we have 3820 crop alive, 664 crops damaged due to other case and 115 crops damaged due to pesticides.

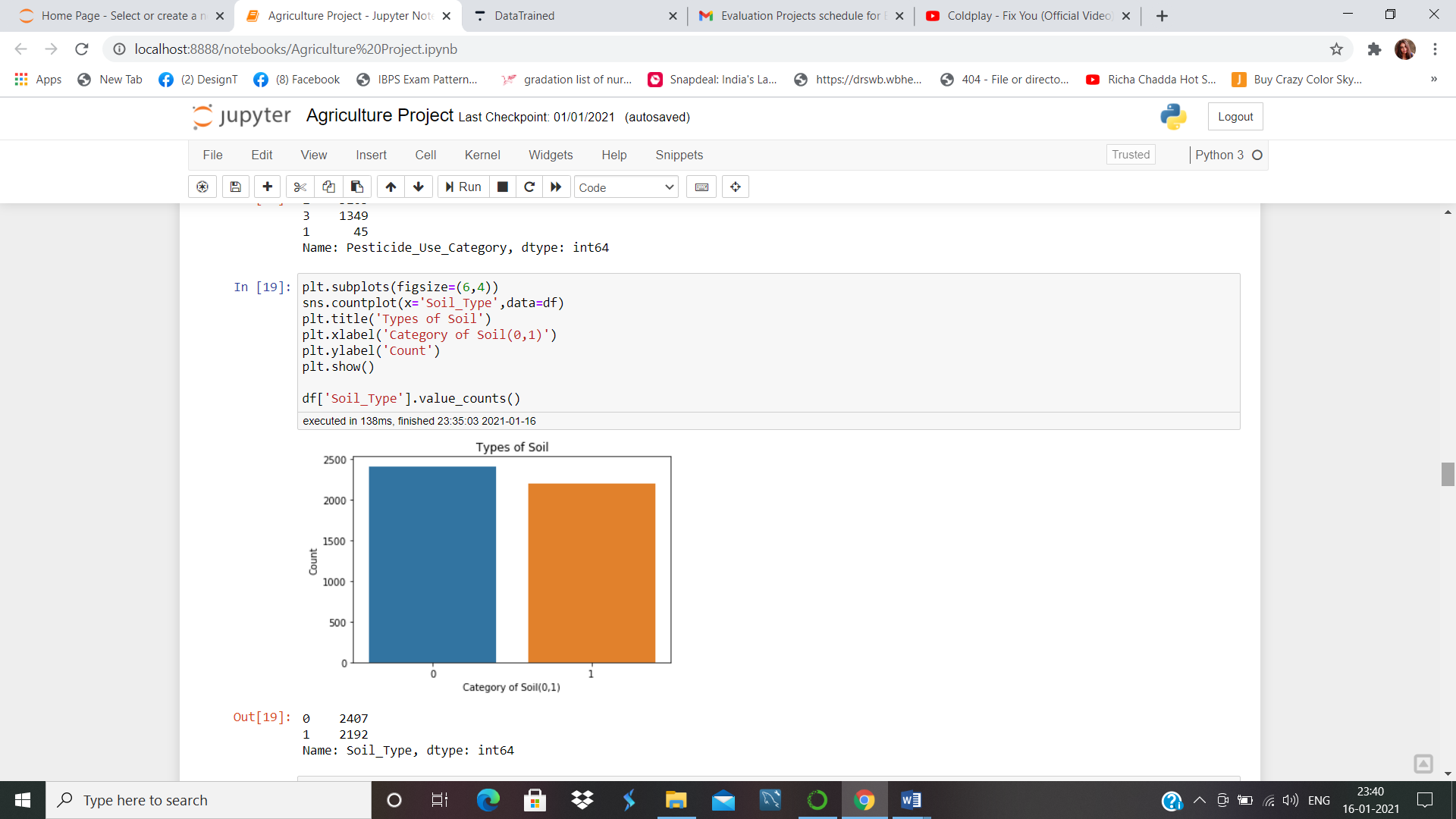
Most of the crops are alive only few got damaged for over doses of pesticides.



There are Two types of crops 0 type of crops are 3565 in count and 1 type of crop is 1034 in count.

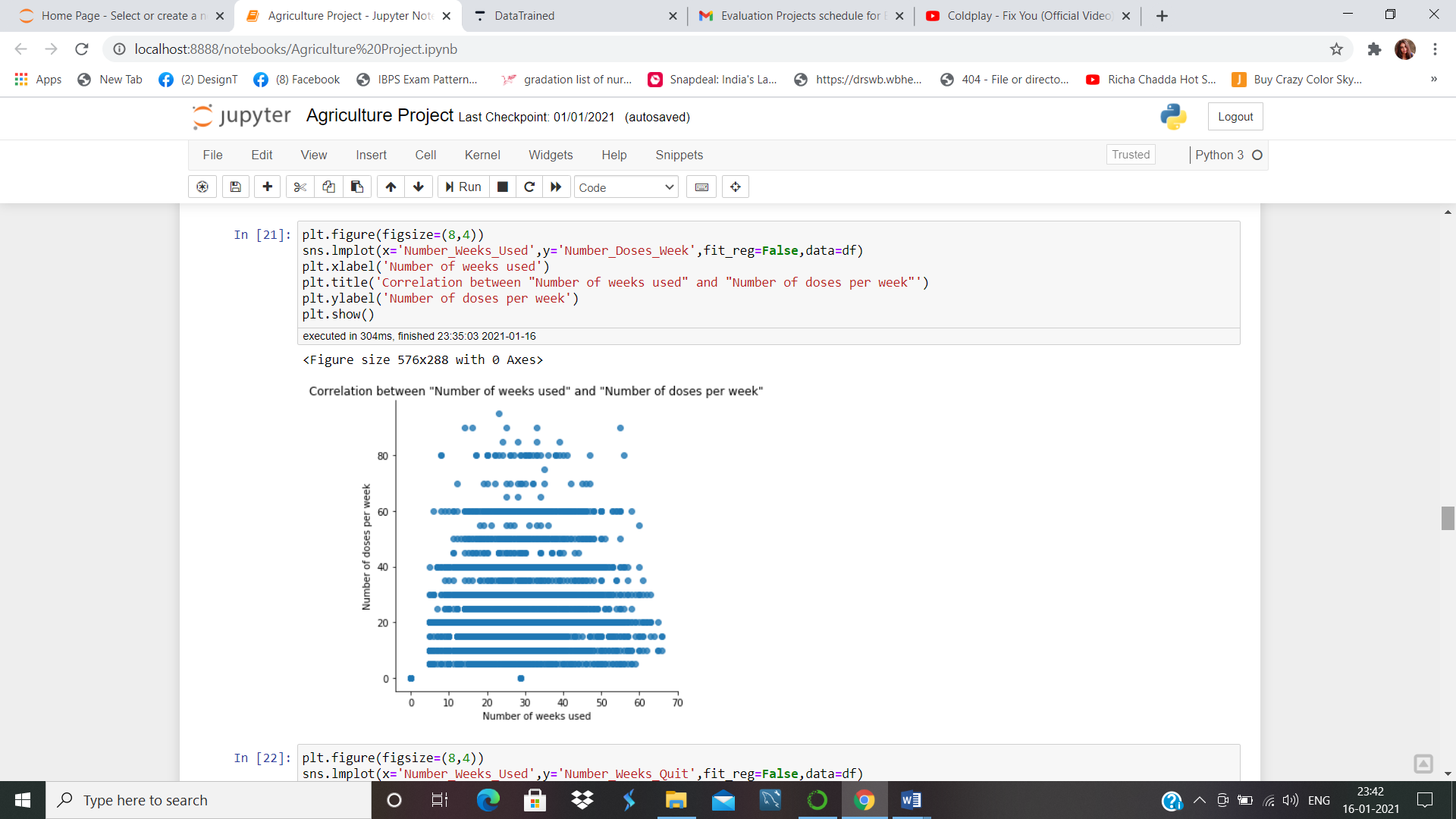


There are only 45 crops where pesticides have never been used. 3205 crops are there, where pesticides have been used previously and 1349 crops are there, where pesticides are being used. So that does mean using of pesticides in crops have reduced drastically. People used to use pesticides in crops in huge amount but now its been slow down.



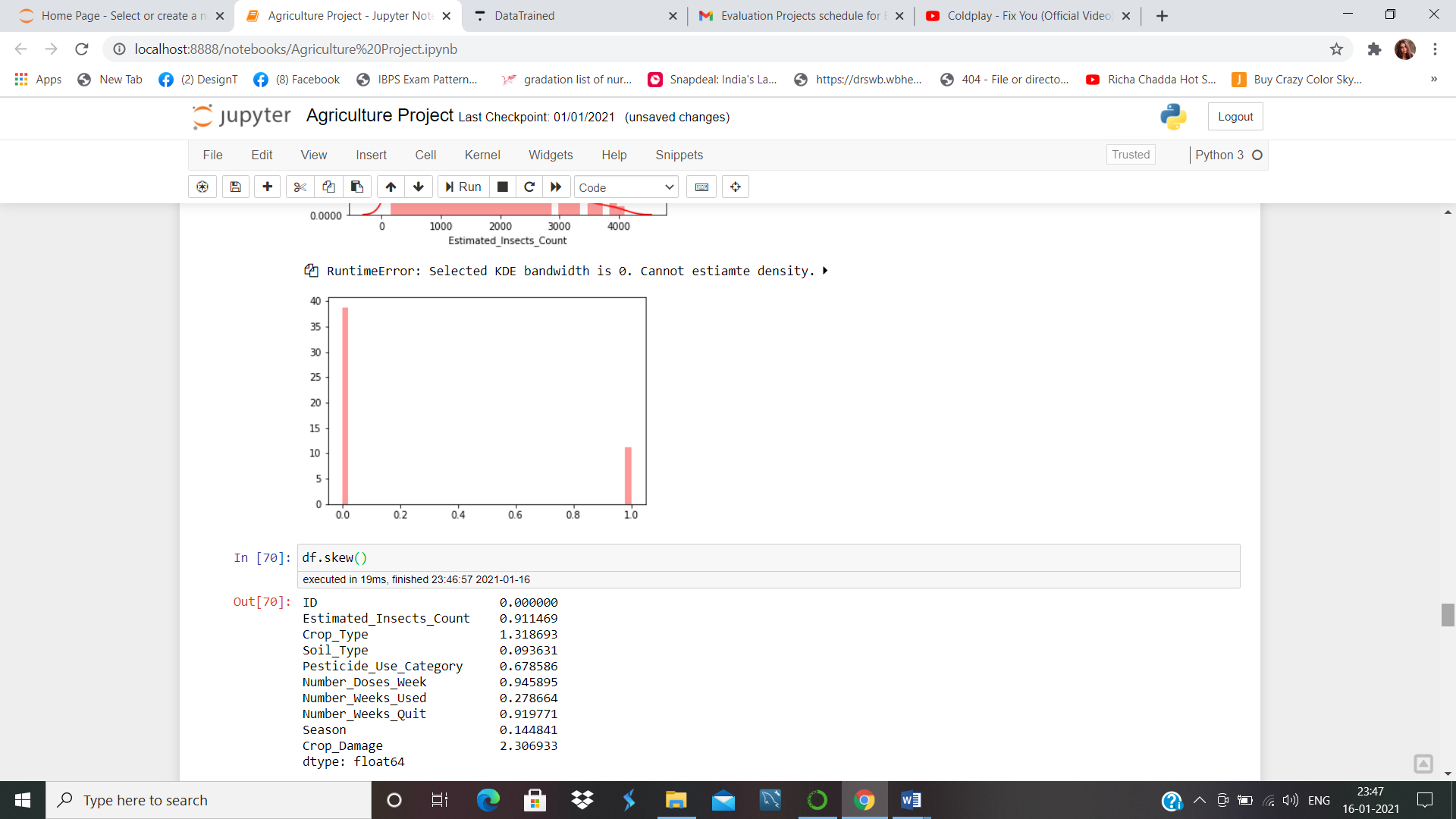
There two soil type in use and both are almost same in number.

We’ll see how how the number of doses per week and number of weeks used are related.

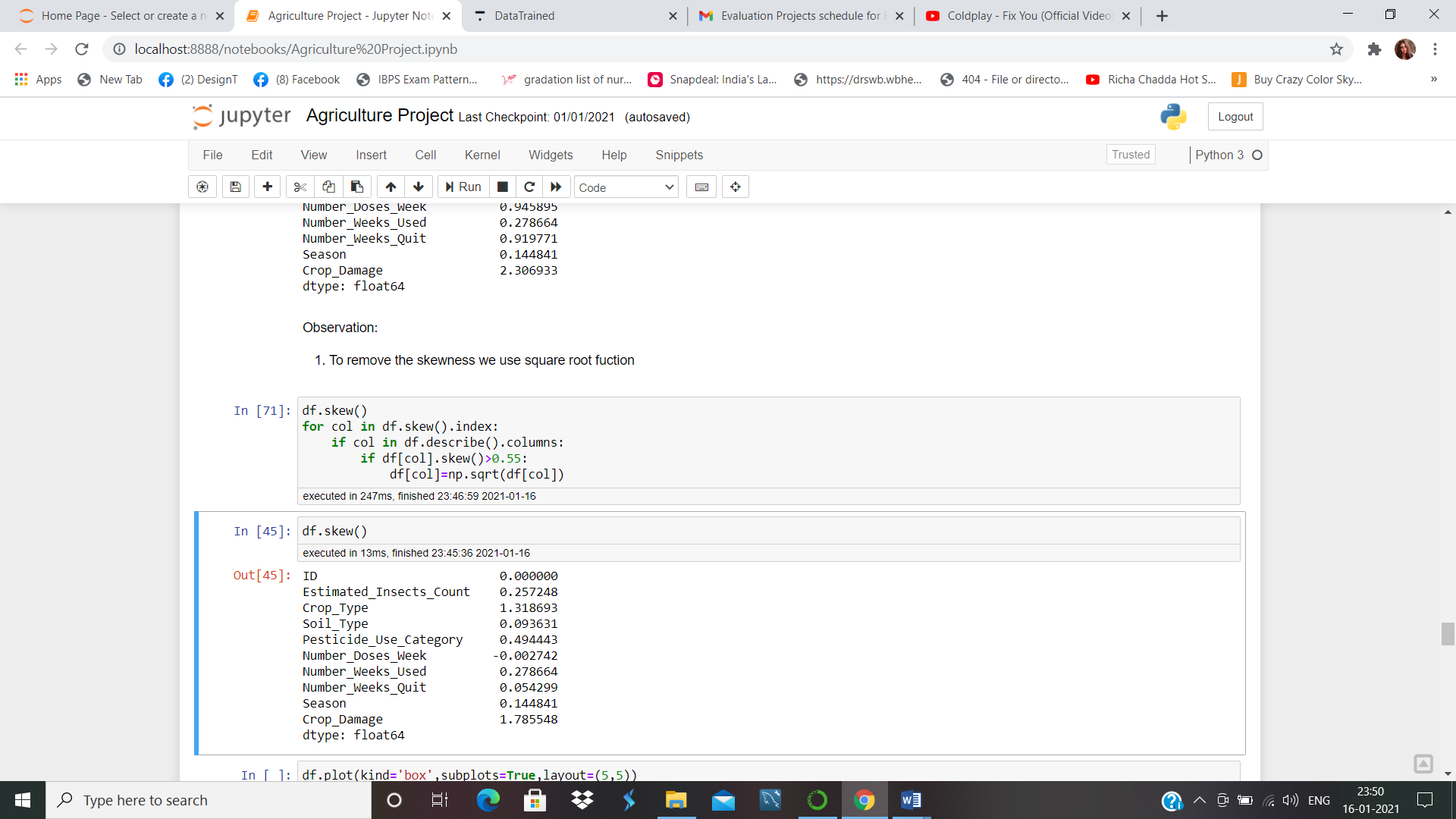


The first few weeks the number doses of pesticides are huge in amount, With time the uses of pesticides gets reduced as we can see in the plot.

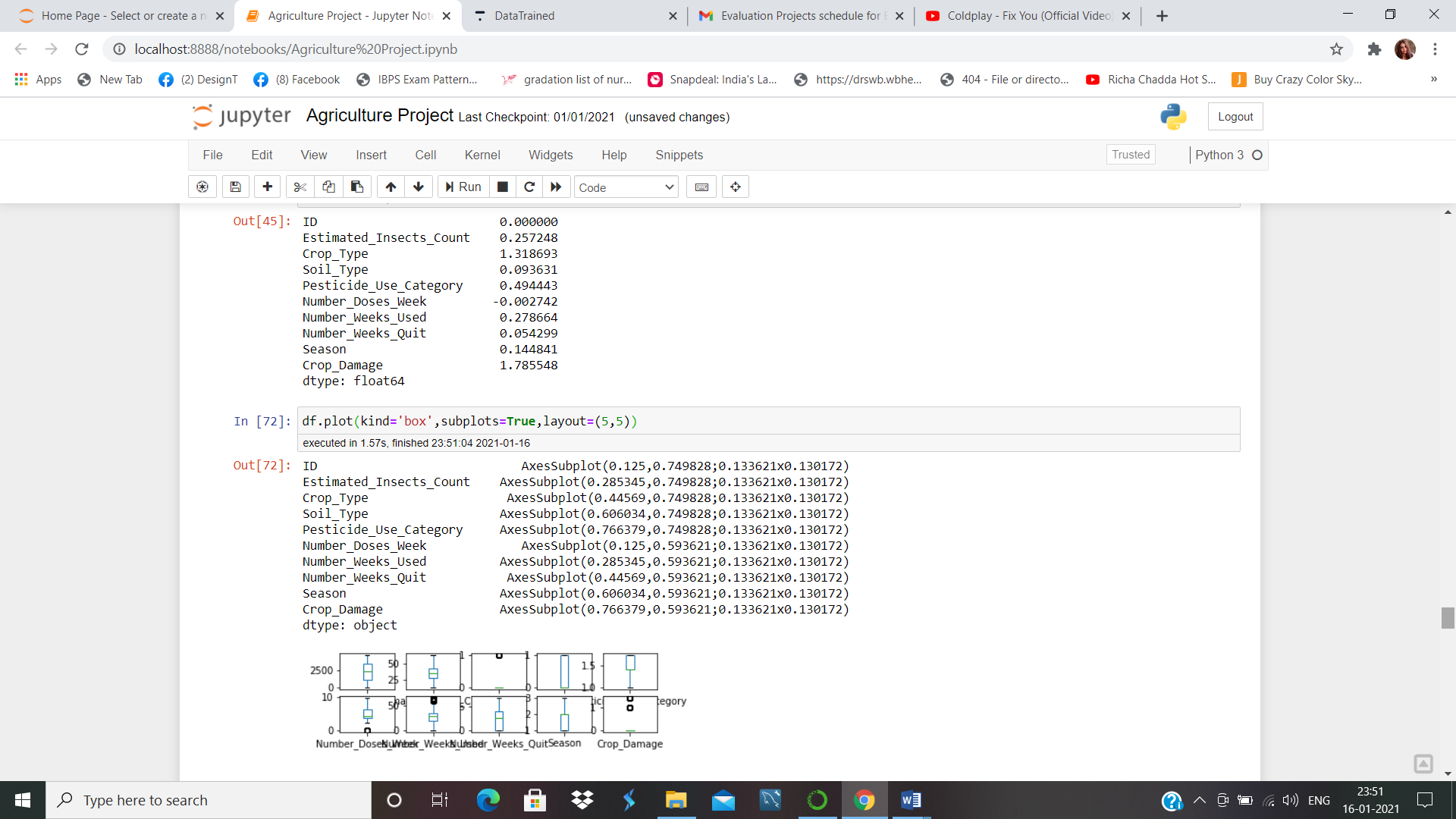
We’ll check the skewness of the data.



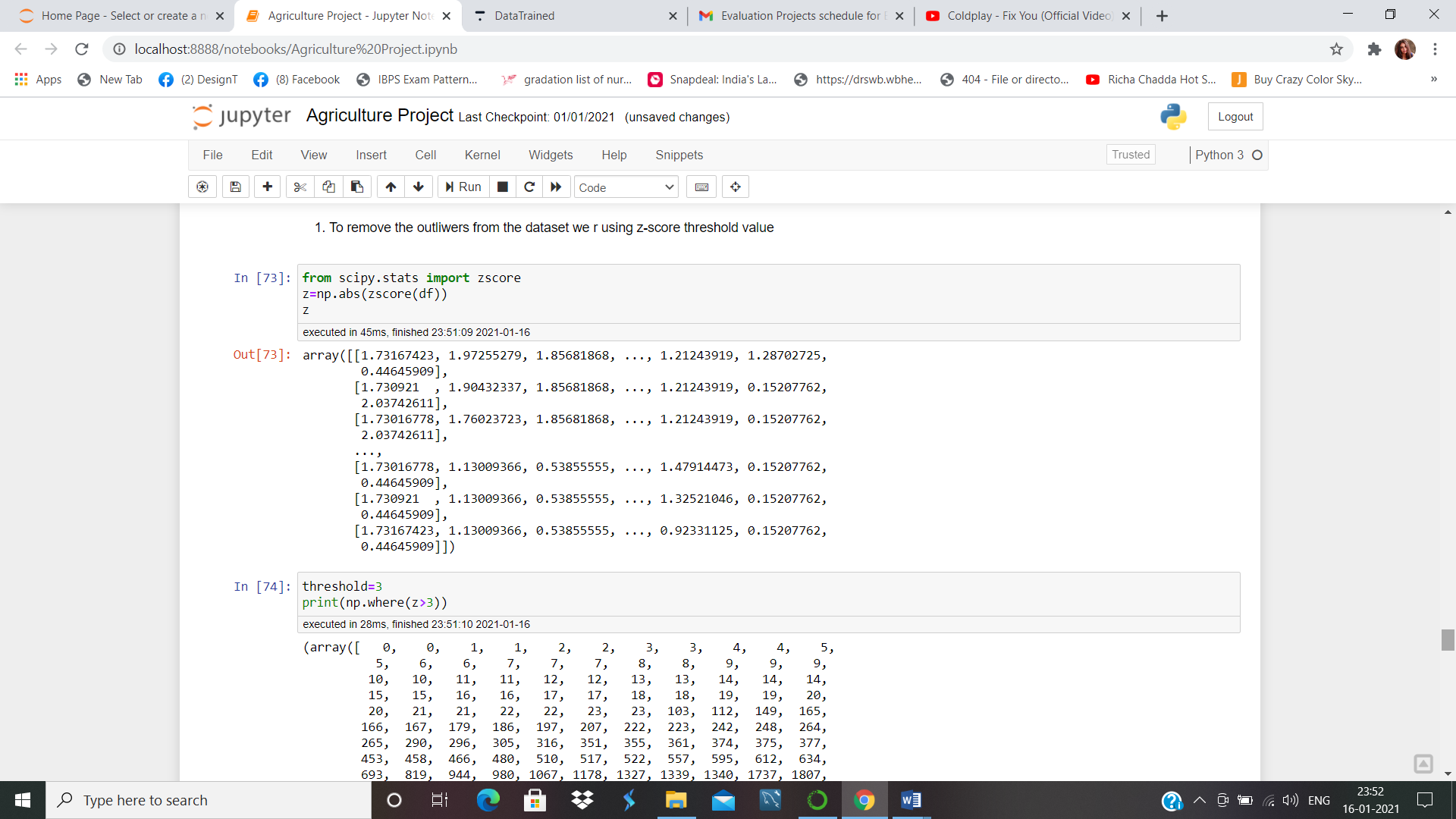
We will remove the skewness by square root function.

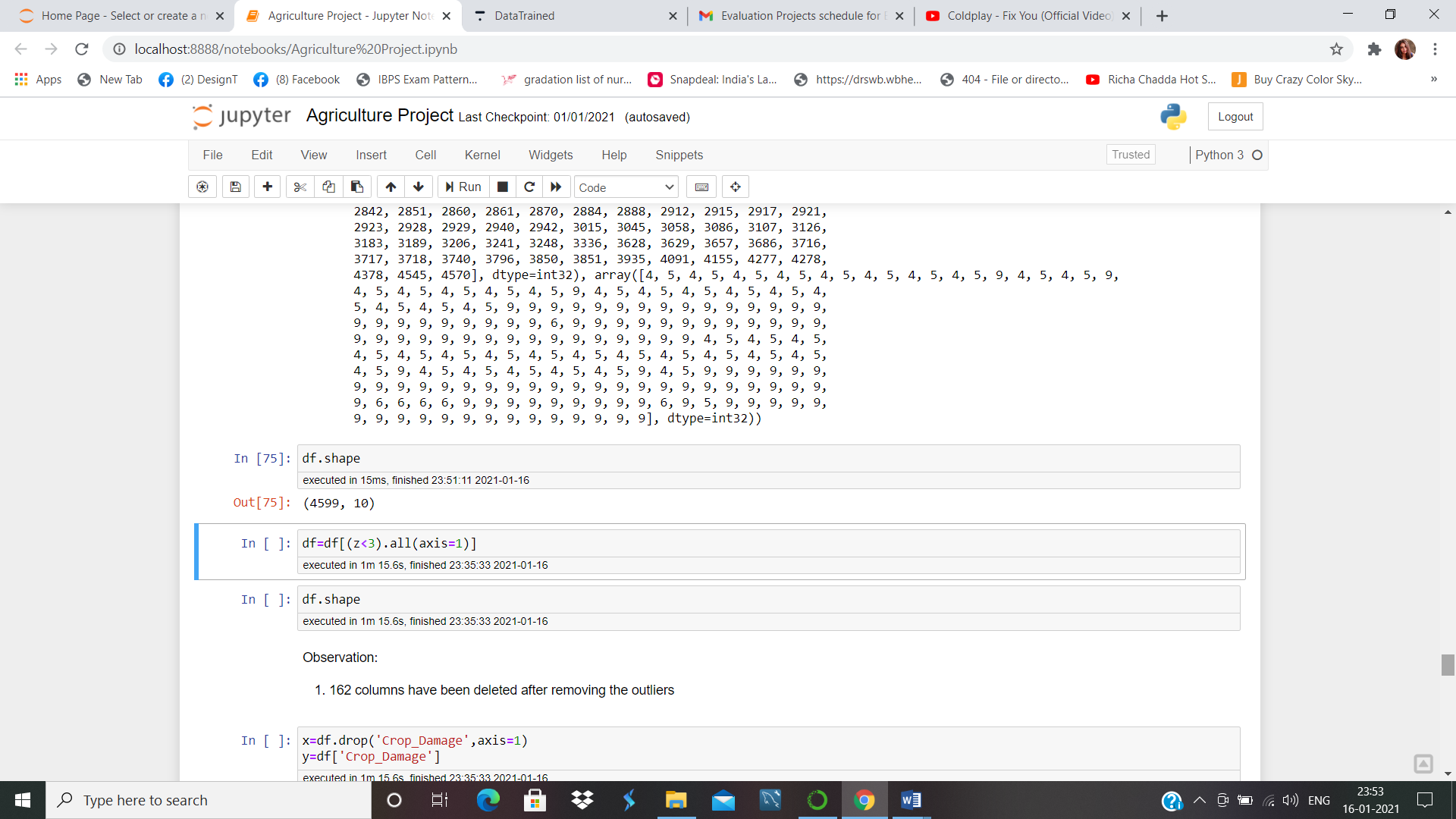


We’ll check the outliers is present in the data or not

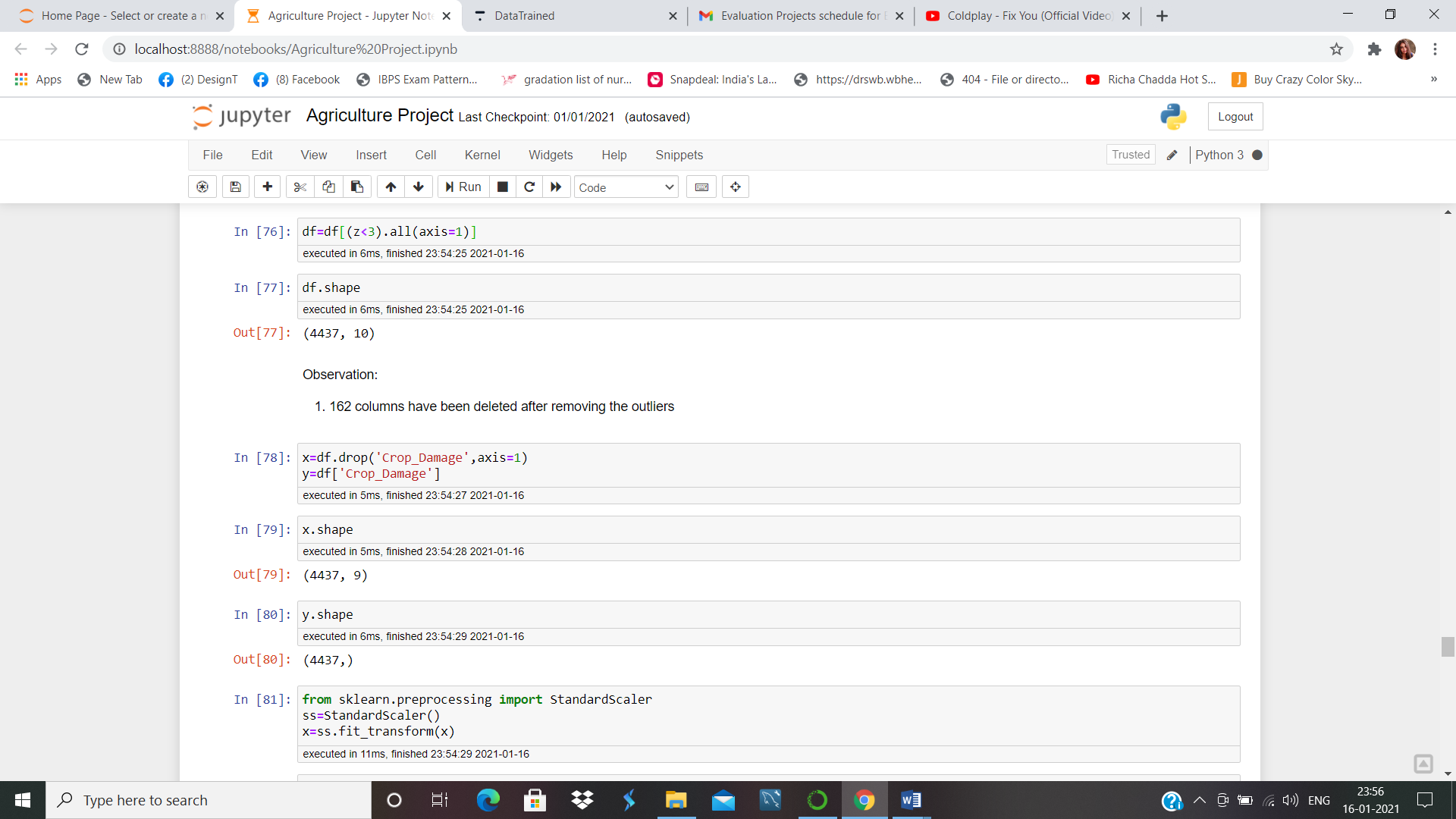


We can remove the outliers by z-score





Now our data is ready for splitting and we can train our model for machine Learning.



We split our model in two DataFrames X and Y. And Scale our X DataFrame with StandardScaler.

Now we split our model in training and testing Dataset and can use our Machine Learning Models.

**from** **sklearn.model\_selection** **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.2,random\_state=42,stratify=y)

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.neighbors** **import** KNeighborsClassifier

**from** **sklearn.tree** **import** DecisionTreeClassifier

**from** **sklearn.svm** **import** SVC

**from** **sklearn.naive\_bayes** **import** GaussianNB

**from** **sklearn.metrics** **import** accuracy\_score

**from** **sklearn.metrics** **import** classification\_report,confusion\_matrix

**from** **sklearn.model\_selection** **import** GridSearchCV,cross\_val\_score

**from** **sklearn.ensemble** **import** RandomForestClassifier

**from** **sklearn.ensemble** **import** AdaBoostClassifier

**from** **sklearn.ensemble** **import** GradientBoostingClassifier

**from** **sklearn.ensemble** **import** BaggingClassifier

model=[LogisticRegression(),GaussianNB(),SVC(),DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier(),AdaBoostClassifier(),GradientBoostingClassifier(),BaggingClassifier()]

**for** m **in** model:

m.fit(x\_train,y\_train)

m.score(x\_train,y\_train)

predm=m.predict(x\_test)

print('Accuracy score of',m,'is:')

print(accuracy\_score(y\_test,predm))

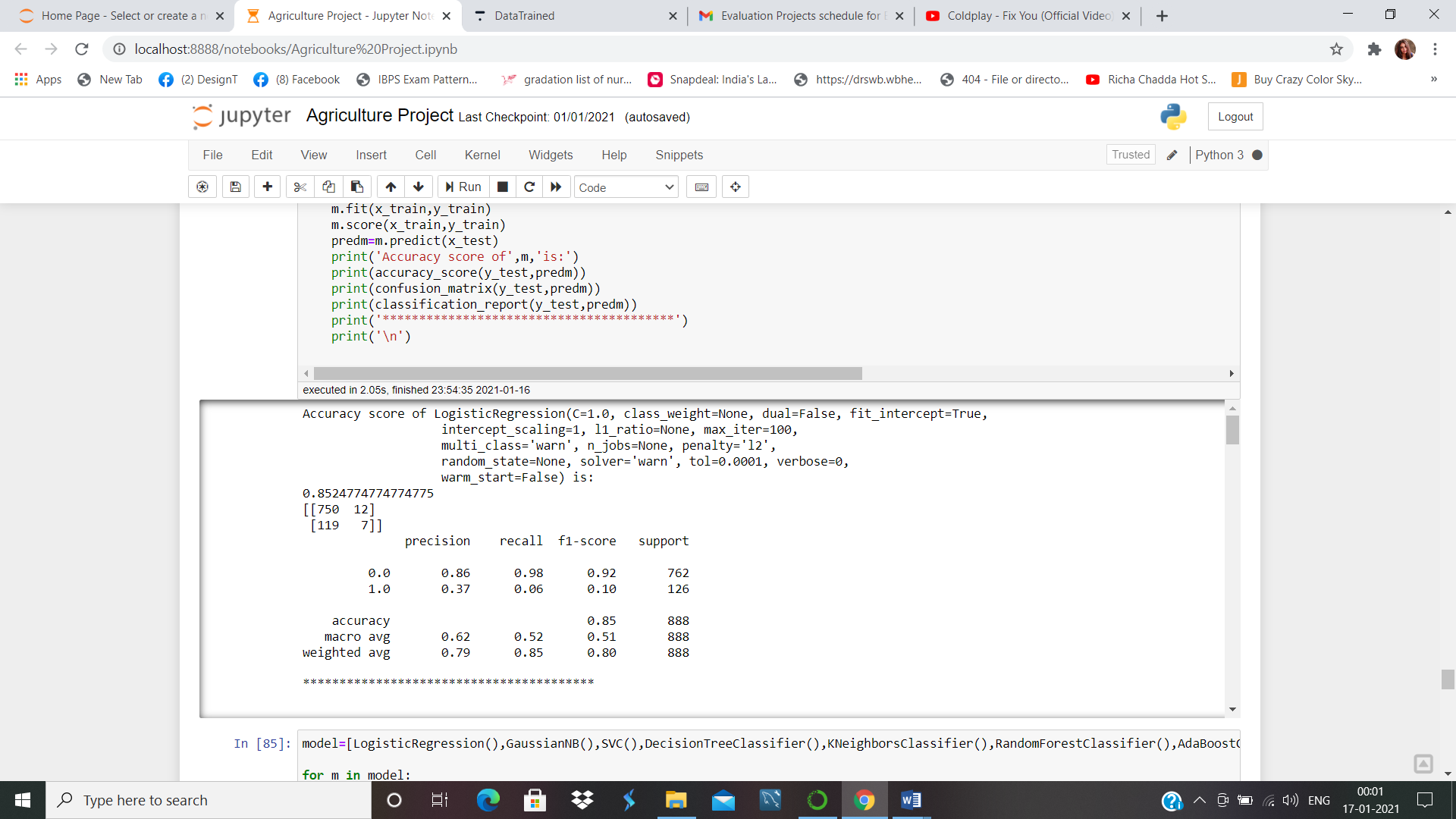
print(confusion\_matrix(y\_test,predm))

print(classification\_report(y\_test,predm))

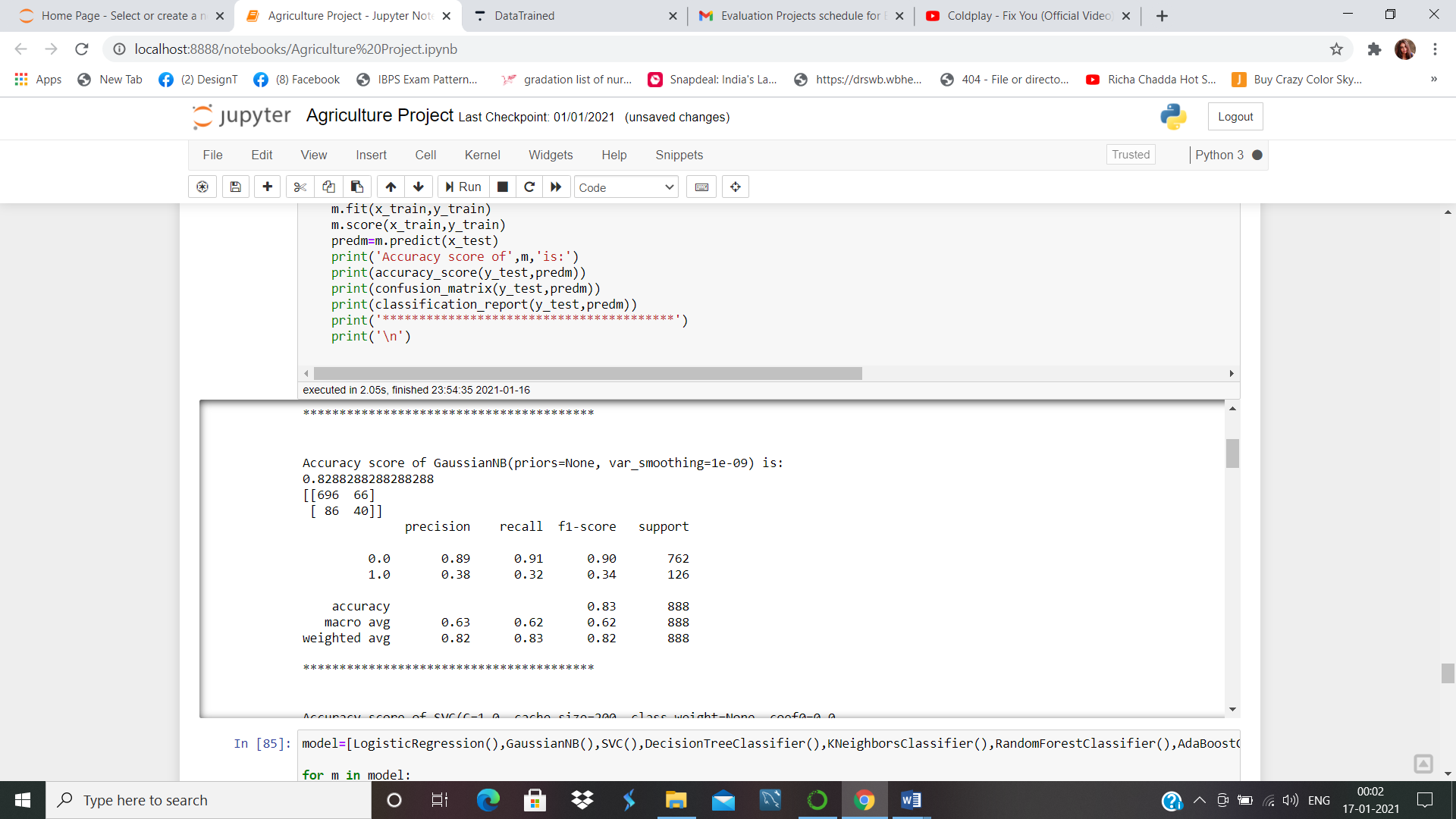
print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print('**\n**')

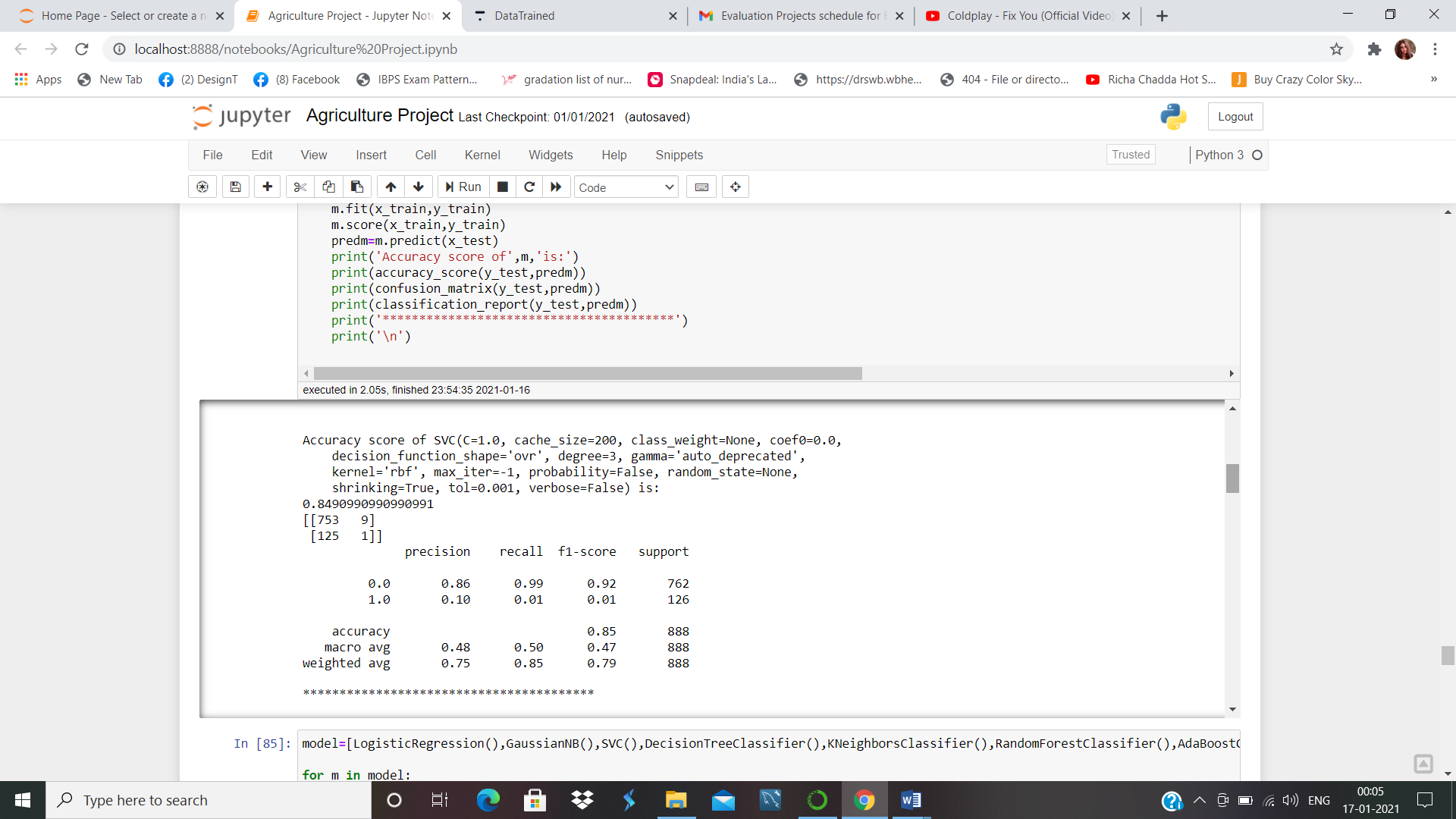
Logistic Regression:-



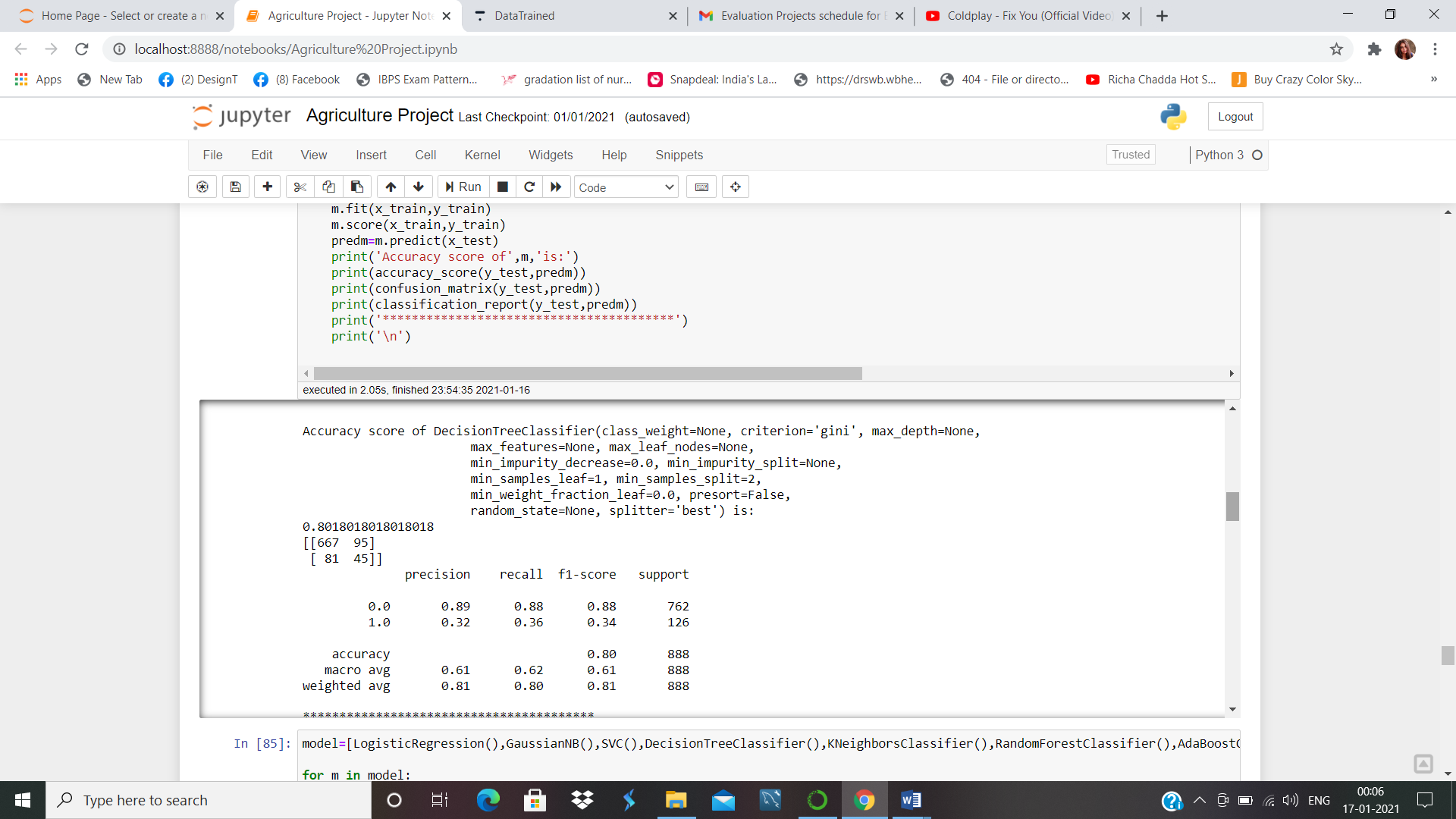
Gaussian Naïve Bayes:-



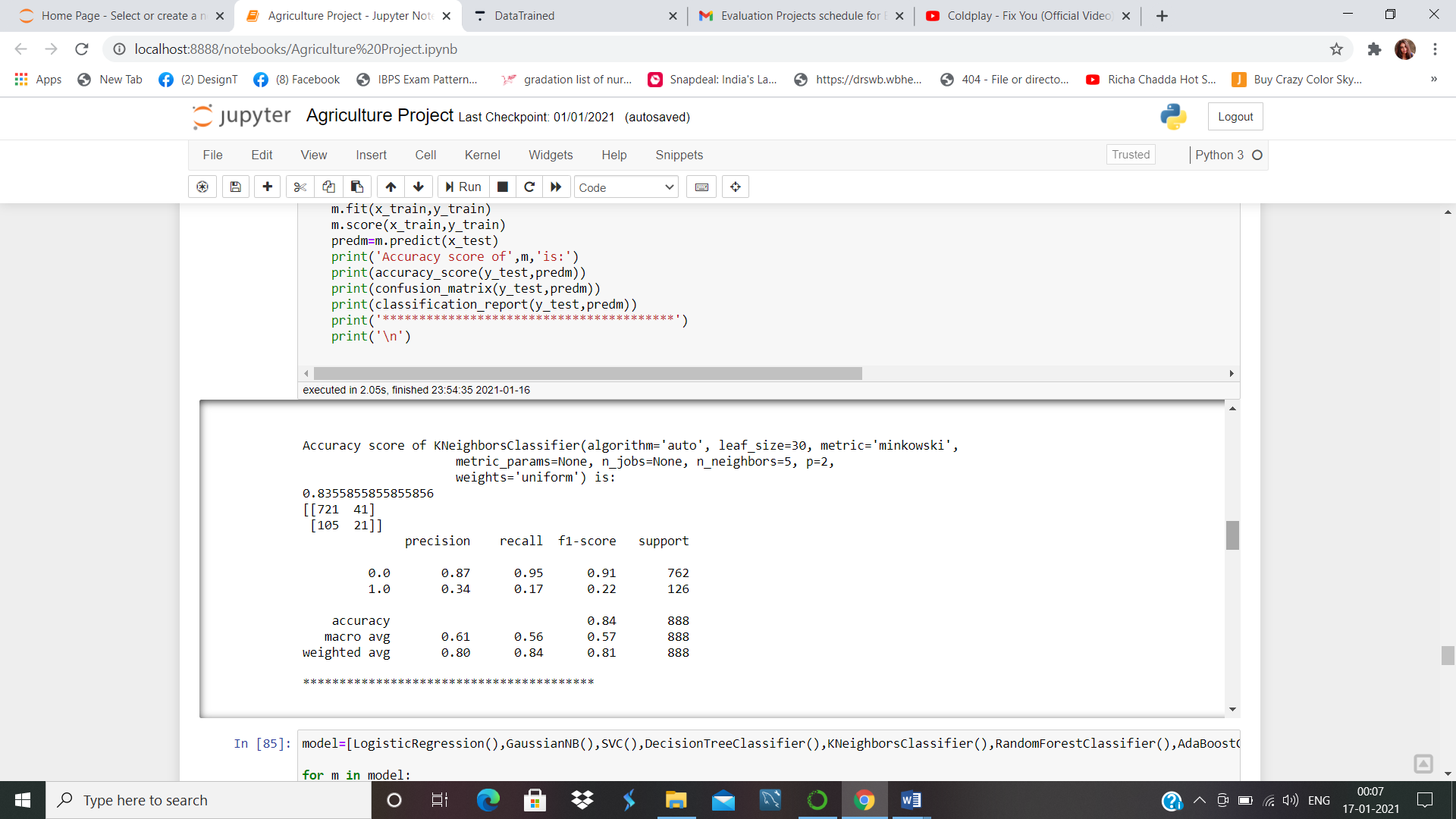
Support Vector Machine:-



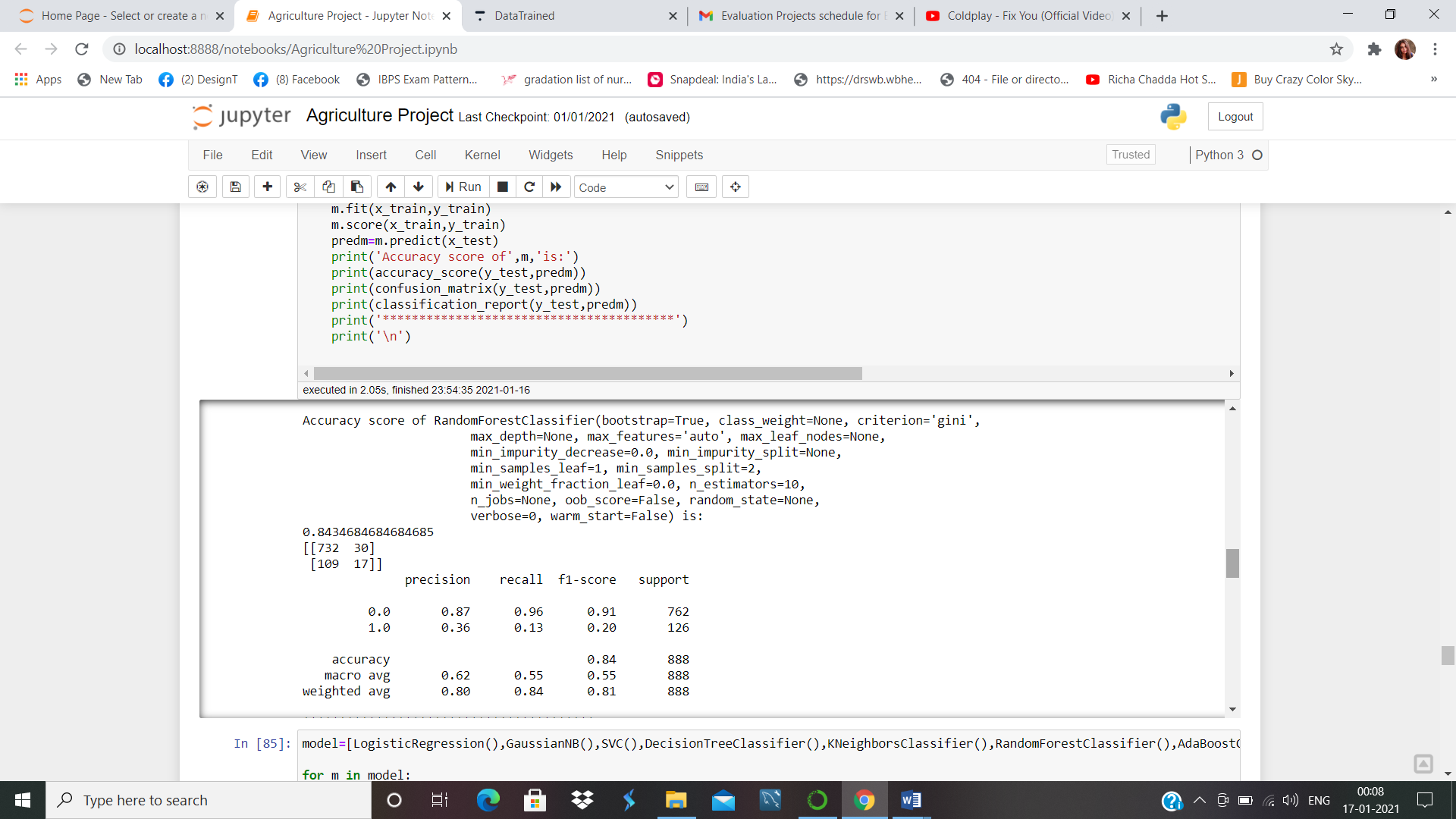
Decission Tree Classifier:-



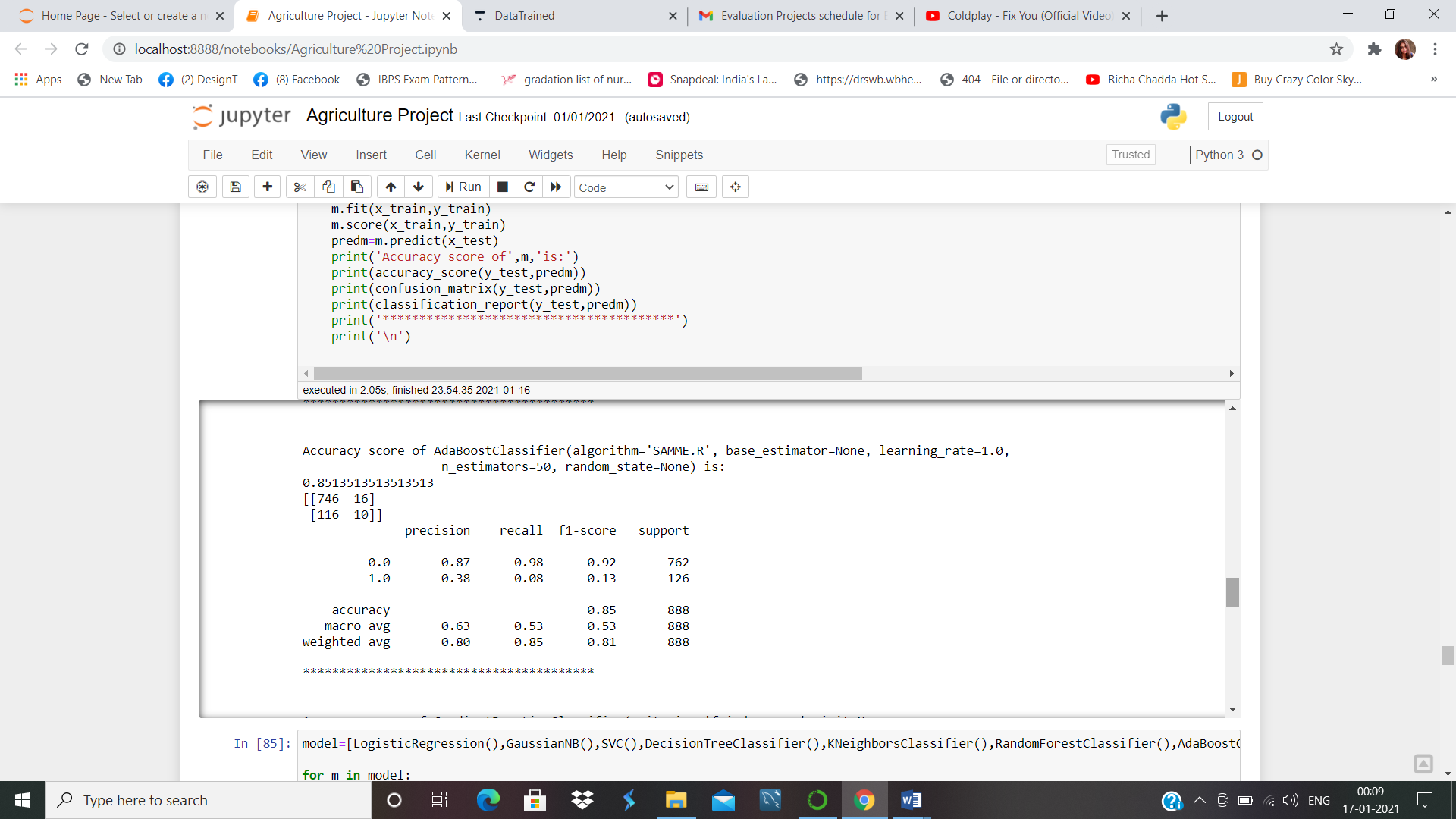
**K Nearest Neighbor:-**



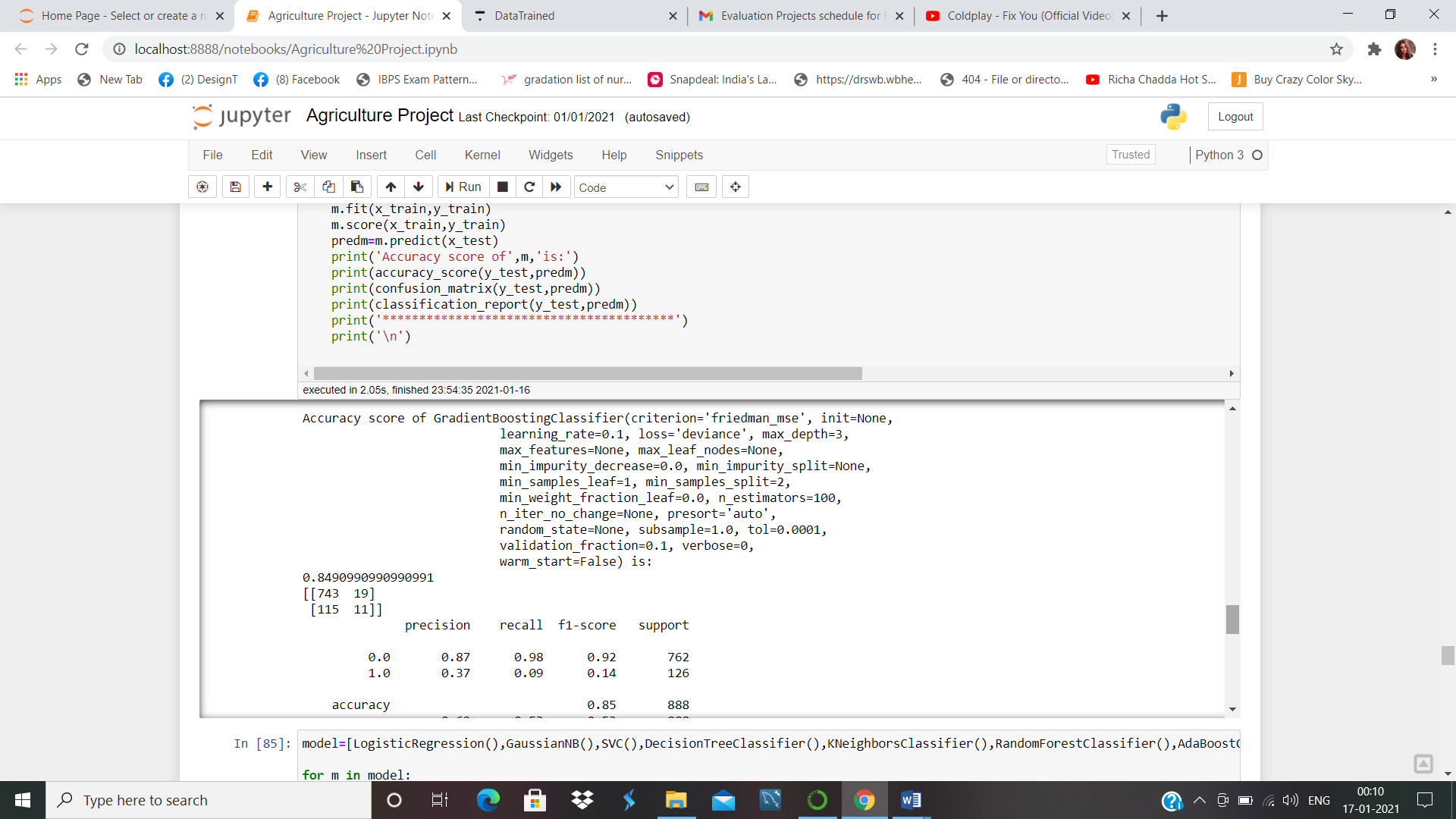
**Random Forrest Classifier:-**



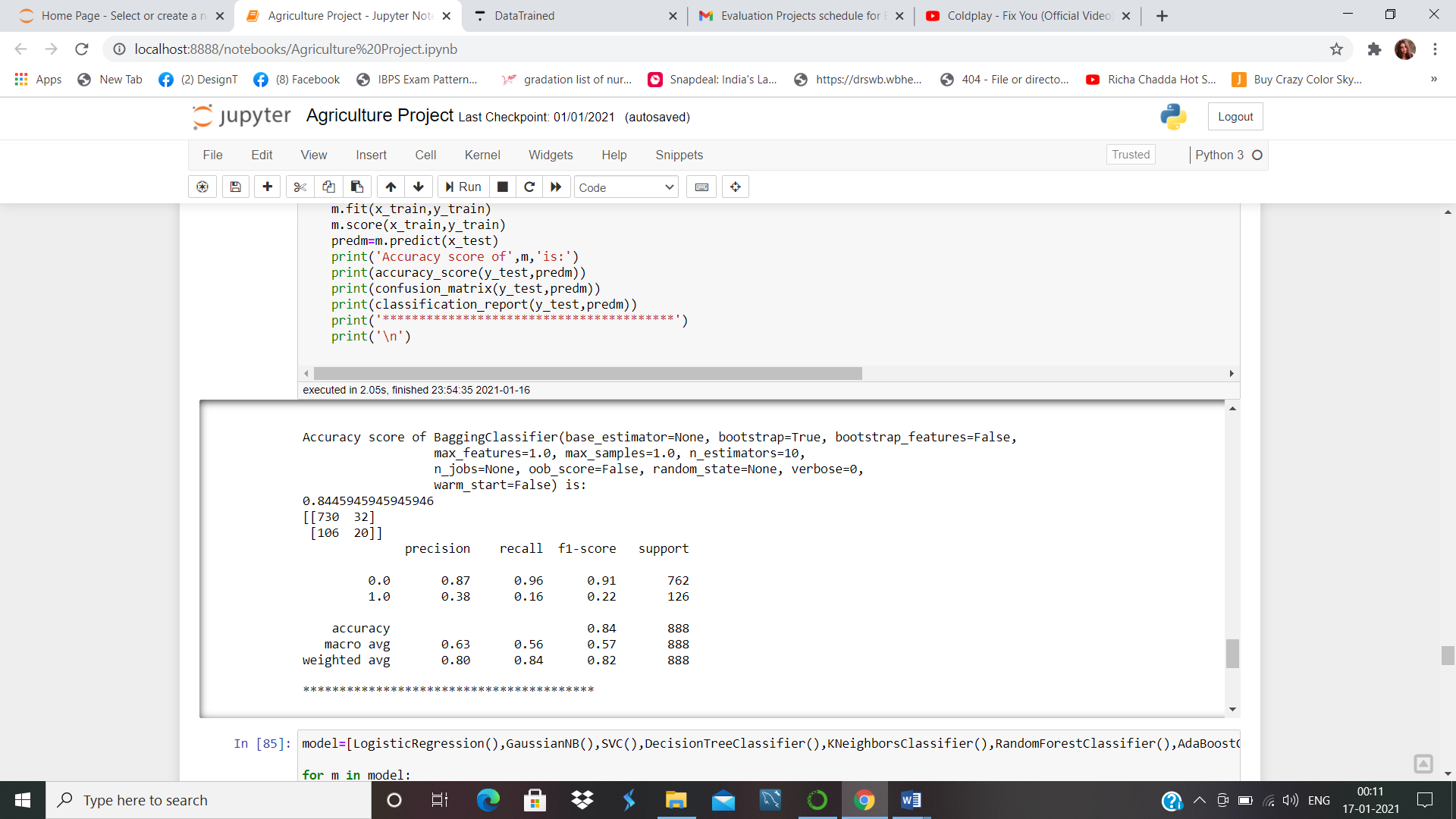
**Adaboost classifier:-**



**Gradient Boosting Classifier:-**

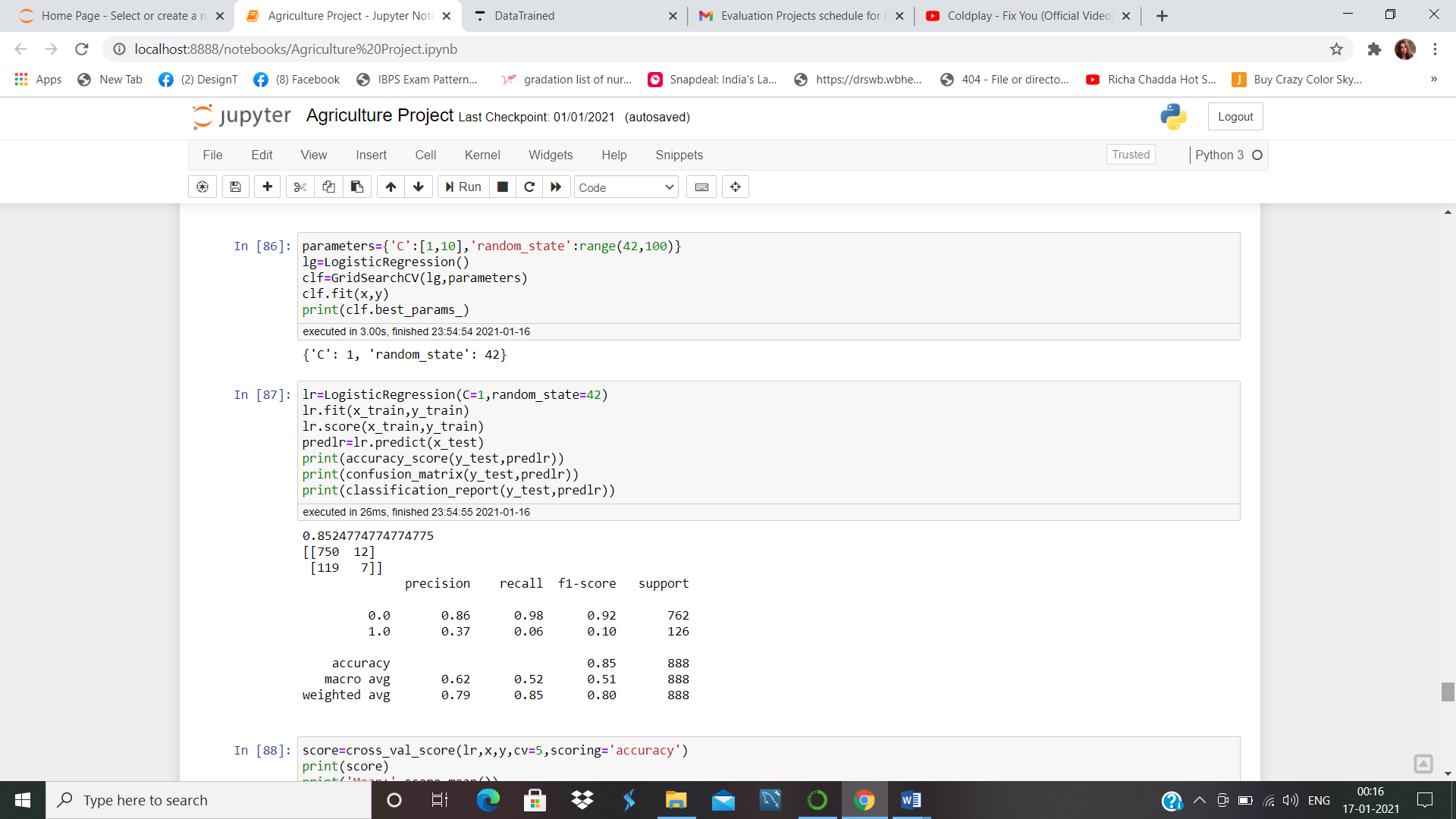


**Bagging Classifier:-**



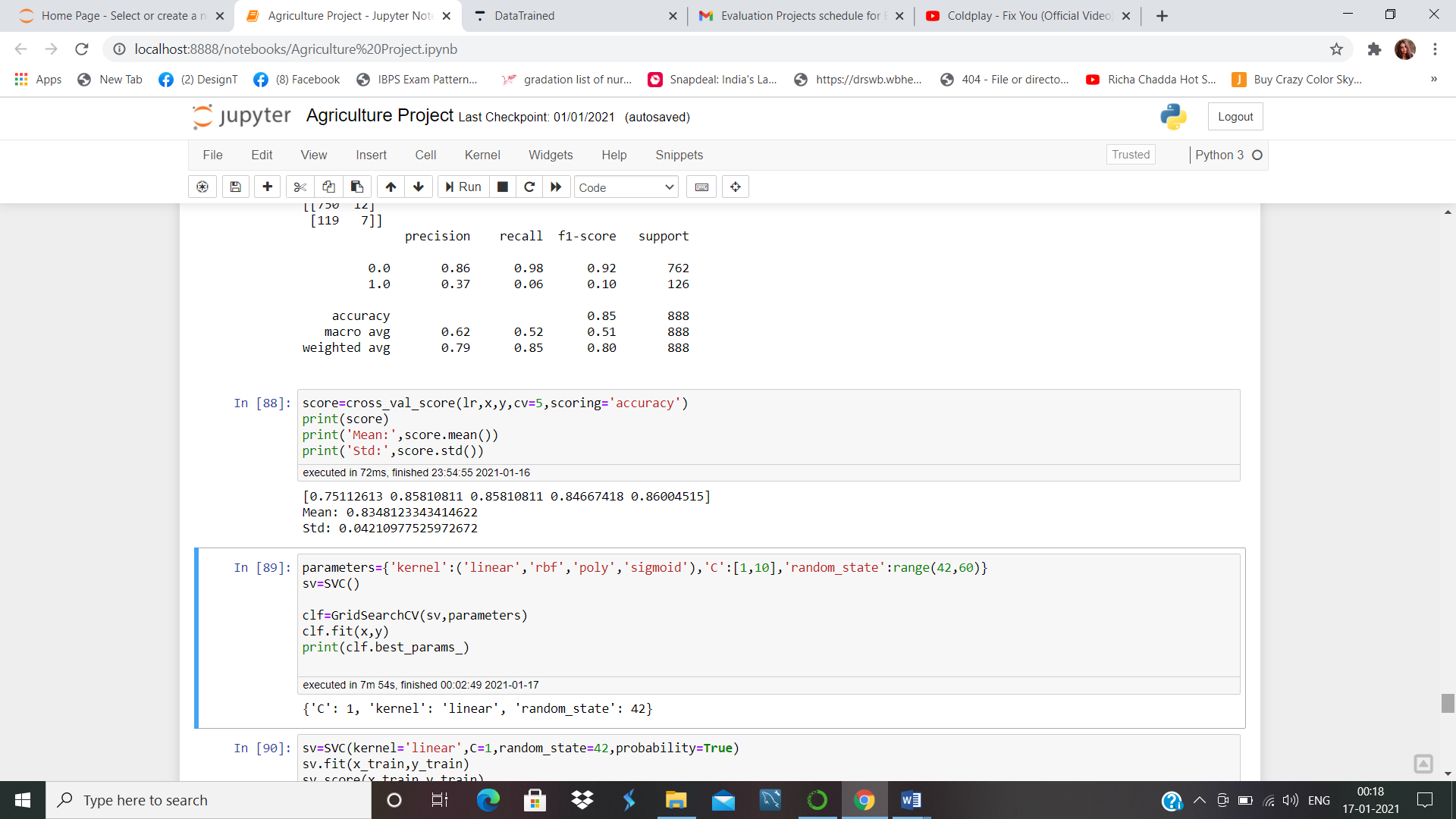
|  |  |
| --- | --- |
| Machine Learning Model | Accuracy |
| Logistic Regression | 85% |
| Gaussian Naïve Bayes | 82% |
| Support Vector Machine | 84% |
| Decission Tree Classifier | 80% |
| **K Nearest Neighbor** | 83% |
| Random Forrest Classifier | 84% |
| Adaboost classifier | 85% |
| Gradient Boosting Classifier | 84% |
| Bagging Classifier | 84% |

Logistic Regression and Adaboost Classifier is working best. We take Logistic Regression and try to find the best parameter and we will check the cross validation score for this particular model as well



K-Fold Cross Validation randomly splits the training data into **K subsets called folds**. Let’s image we would split our data into 4 folds (K = 4). Our random forest model would be trained and evaluated 4 times, using a different fold for evaluation everytime, while it would be trained on the remaining 3 folds.

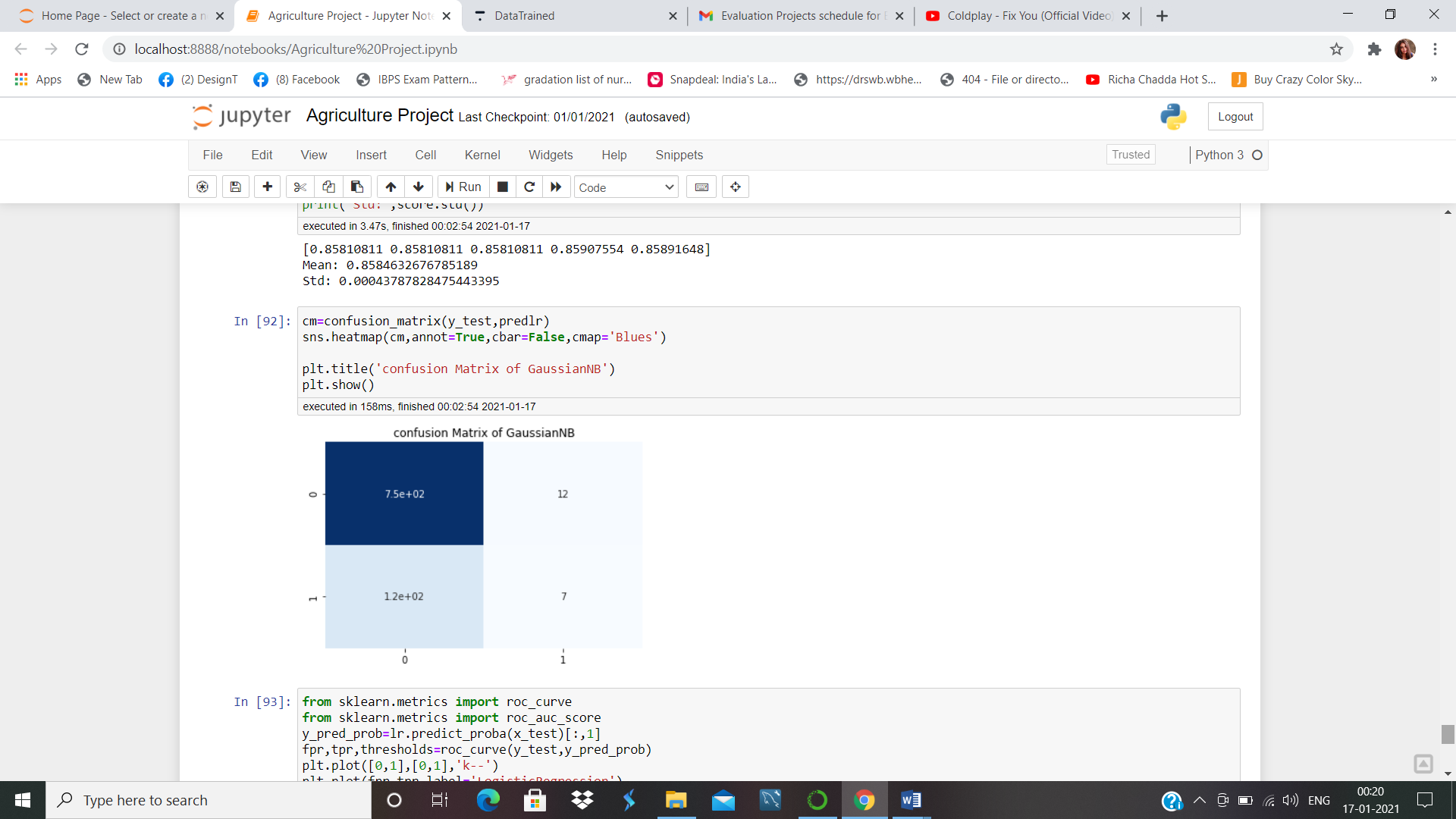
We will do the Cross-Validation and use 5 folds for this particular model-



It shows our average accuracy is 83% with standard deviation of 4%

This shows our accuracy of the model can differ from +-4%.

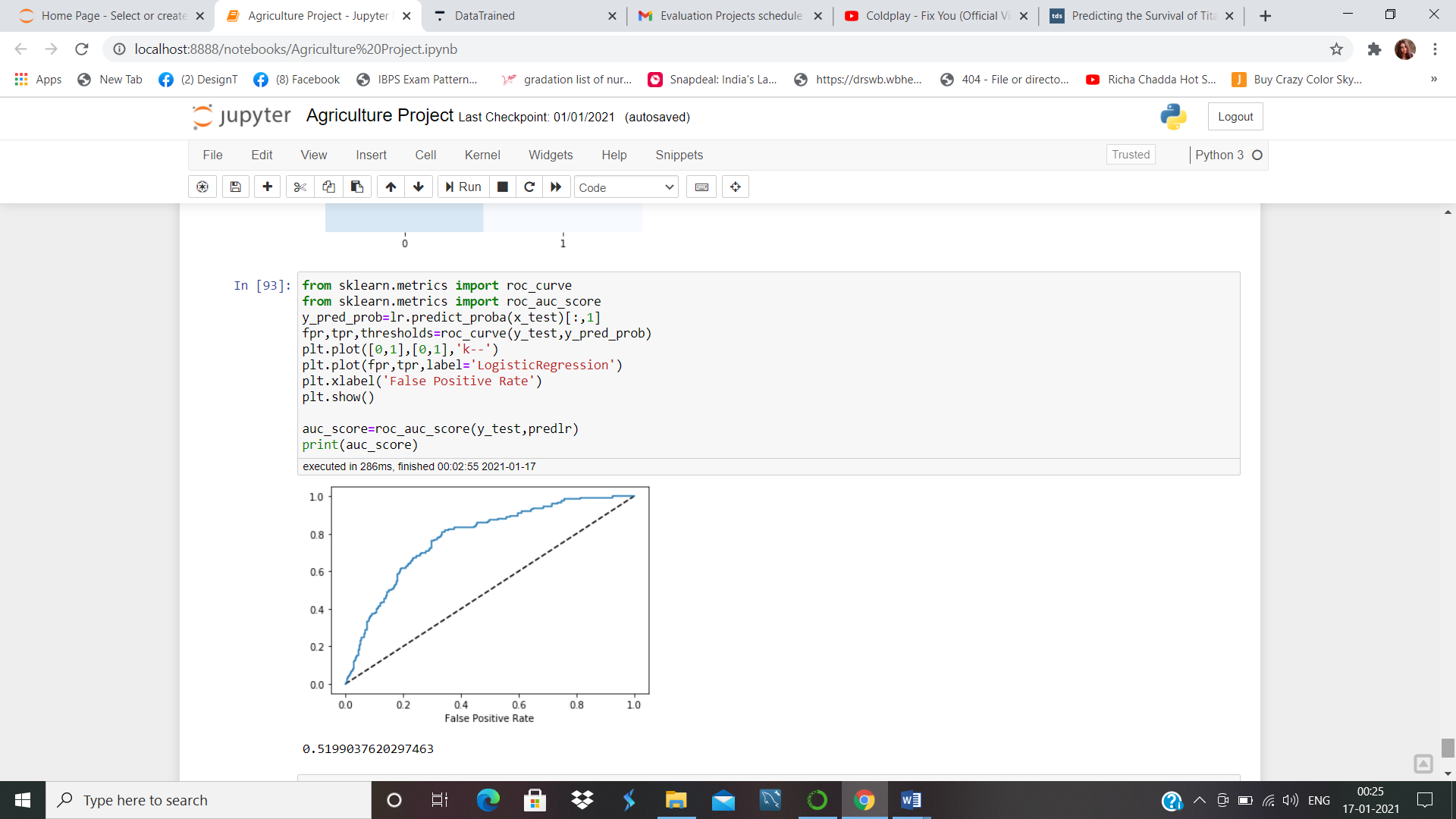
We’ll see the confusion matrix of this dataset with GaussianNb



This shows the 1st row that crop is alive is predicted wrong 12 times and the crop is damaged predicted wrong as alive 7 times.

We can check the AUC-ROC Curve:-

Another way to evaluate and compare your binary classifier is provided by the ROC AUC Curve. This curve plots the true positive rate (also called recall) against the false positive rate (ratio of incorrectly classified negative instances), instead of plotting the precision versus the recall.



The dotted line in the middle represents a purely random classifier and therefore your classifier should be as far away from it as possible. Our model seems to do a good job.Of course we also have a trade off here, because the classifier produces more false positives, the higher the true positive rate is.

There is place to improve our model by doing several other data preprocessing. But our model seems to work fine with 85% accuracy.