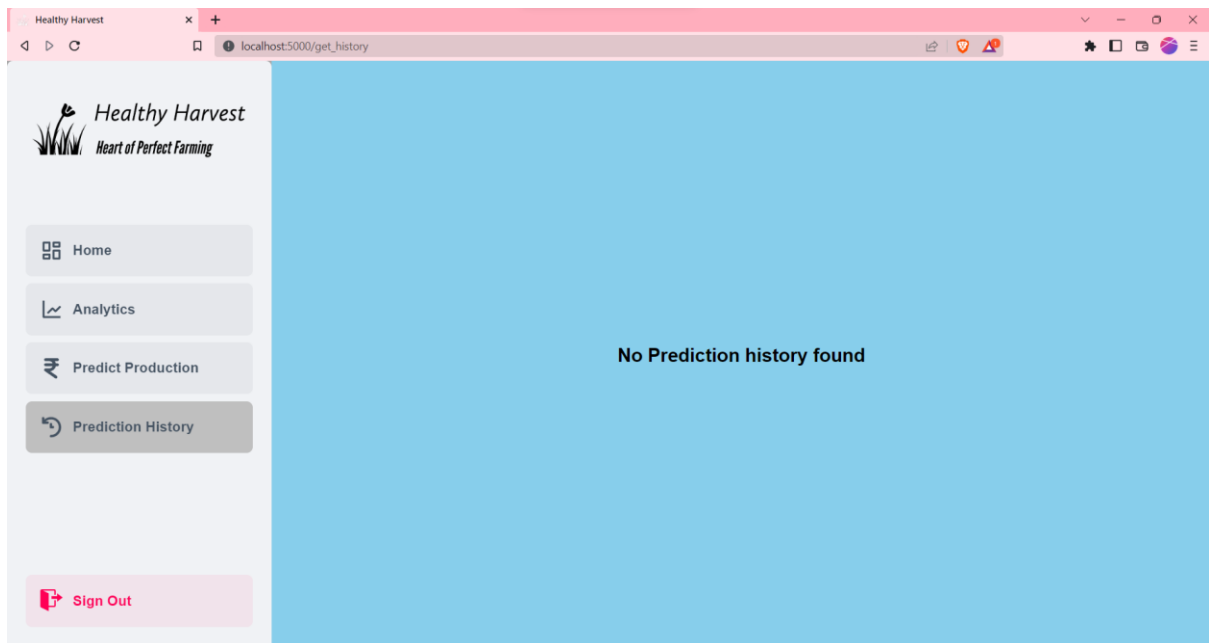


Functional Requirement (Epic): Dashboard**User Story Number:** USN-7**User Story / Task Story:** As a user, I can view the previous results of predictions done by me.**Points:** 2**Priority:** Low**Team Members:** Marieswari M**Screenshots:**

The screenshot shows the 'Healthy Harvest' dashboard with a table of prediction history. The table has columns: State, District, Crop Year, Season, Crop, Area, and Production(In Tons). The table contains three rows of data. Above the table are five filter buttons: 'Filter State', 'Filter District', 'Filter Crop Year', 'Filter Season', and 'Filter Crop'. The left sidebar is identical to the previous screenshot.

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Tamil Nadu	MADURAI	2022	Whole Year	Cabbage	20	1582.4
Andhra Pradesh	ANANTAPUR	2023	Winter	Beet Root	100	0
Tamil Nadu	COIMBATORE	2023	Winter	Tea	150	1709.52

Healthy Harvest
Heart of Perfect Farming

Home Analytics Predict Production Prediction History Sign Out

Filter State Filter District Filter Crop Year Filter Season Filter Crop

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Andhra Pradesh	ANANTAPUR	2023	Winter	Beet Root	100	0
Tamil Nadu	MADURAI	2022	Whole Year	Cabbage	20	1582.4
Tamil Nadu	COIMBATORE	2023	Winter	Tea	150	1709.52

Healthy Harvest
Heart of Perfect Farming

Home Analytics Predict Production Prediction History Sign Out

Tamil Filter District Filter Crop Year Filter Season Filter Crop

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Tamil Nadu	MADURAI	2022	Whole Year	Cabbage	20	1582.4
Tamil Nadu	COIMBATORE	2023	Winter	Tea	150	1709.52

Healthy Harvest
Heart of Perfect Farming

Home
Analytics
Predict Production
Prediction History
Sign Out

Filter State: Coimbatore Filter Crop Year: Filter Season: Filter Crop:

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Tamil Nadu	COIMBATORE	2023	Winter	Tea	150	1709.52

Healthy Harvest
Heart of Perfect Farming

Home
Analytics
Predict Production
Prediction History
Sign Out

Filter State: Filter District: 2023 Filter Season: Filter Crop:

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Andhra Pradesh	ANANTAPUR	2023	Winter	Beet Root	100	0
Tamil Nadu	COIMBATORE	2023	Winter	Tea	150	1709.52

Healthy Harvest
Heart of Perfect Farming

Filter State Filter District Filter Crop Year Whole Filter Crop

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Tamil Nadu	MADURAI	2022	Whole Year	Cabbage	20	1582.4

Home Analytics Predict Production Prediction History Sign Out

Healthy Harvest
Heart of Perfect Farming

Filter State Filter District Filter Crop Year Filter Season Cab

State	District	Crop Year	Season	Crop	Area	Production(In Tons)
Tamil Nadu	MADURAI	2022	Whole Year	Cabbage	20	1582.4

Home Analytics Predict Production Prediction History Sign Out

Functional Requirement (Epic): Data Maintenance**User Story Number:** USN-10

User Story / Task Story: As an administrator, I can collect Data and maintain it and update whenever necessary

Points: 1

Priority: Medium

Team Members: Marieswari M

Screenshots:

The first screenshot shows the initial data loading and inspection. The code imports necessary libraries (numpy, matplotlib, pandas) and reads a CSV file named 'crop_production.csv'. The resulting DataFrame, 'crop_data', has 246091 rows and 7 columns: State_Name, District_Name, Crop_Year, Season, Crop, Area, and Production. A preview of the data shows entries for Andaman and Nicobar Islands and West Bengal.

```
In [1]: # Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

crop_data = pd.read_csv('crop_production.csv')
crop_data
```

Out[1]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Areca nut	1254.0	2000.0
1	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	Andaman and Nicobar Islands	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0
...
246086	West Bengal	PURULIA	2014	Summer	Rice	306.0	801.0
246087	West Bengal	PURULIA	2014	Summer	Sesamum	627.0	463.0
246088	West Bengal	PURULIA	2014	Whole Year	Sugarcane	324.0	16250.0
246089	West Bengal	PURULIA	2014	Winter	Rice	279151.0	597899.0
246090	West Bengal	PURULIA	2014	Winter	Sesamum	175.0	88.0

246091 rows x 7 columns

```
In [2]: crop_data = crop_data.dropna()
crop_data
```

Out[2]:

	State_Name	District_Name	Crop_Year	Season	Crop	Area	Production
0	Andaman and Nicobar Islands	NICOBARS	2000	Kharif	Areca nut	1254.0	2000.0

The second screenshot shows the data being cleaned and reshaped. The 'State_Name' column is dropped, and the data is reshaped into a wide format using 'pd.get_dummies'. The resulting DataFrame, 'dummy', has 242361 rows and 6 columns: Crop_Year, Area, Production, and three dummy variables for Districts (PARAGANAS NORTH, PARAGANAS SOUTH, and ADILABAD). A preview of the data shows entries for PARAGANAS NORTH and PARAGANAS SOUTH.

```
In [3]: data = crop_data.drop(['State_Name'], axis = 1)
data
```

Out[3]:

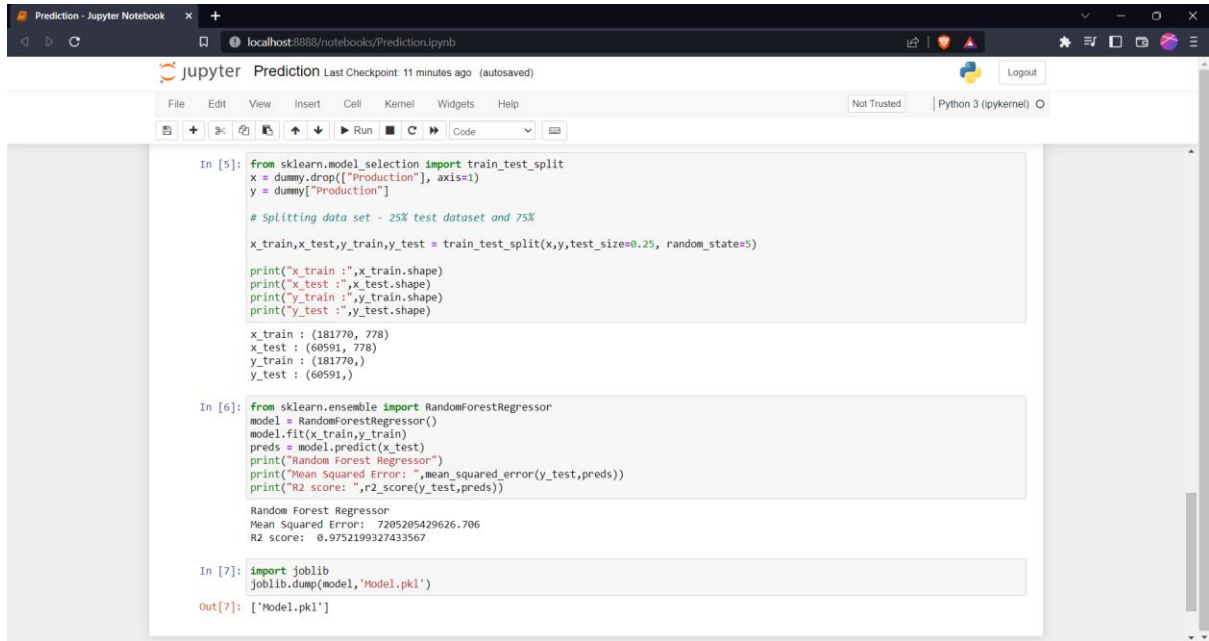
	District_Name	Crop_Year	Season	Crop	Area	Production
0	NICOBARS	2000	Kharif	Areca nut	1254.0	2000.0
1	NICOBARS	2000	Kharif	Other Kharif pulses	2.0	1.0
2	NICOBARS	2000	Kharif	Rice	102.0	321.0
3	NICOBARS	2000	Whole Year	Banana	176.0	641.0
4	NICOBARS	2000	Whole Year	Cashewnut	720.0	165.0
...
246086	PURULIA	2014	Summer	Rice	306.0	801.0
246087	PURULIA	2014	Summer	Sesamum	627.0	463.0
246088	PURULIA	2014	Whole Year	Sugarcane	324.0	16250.0
246089	PURULIA	2014	Winter	Rice	279151.0	597899.0
246090	PURULIA	2014	Winter	Sesamum	175.0	88.0

242361 rows x 6 columns

```
In [4]: dummy = pd.get_dummies(data)
dummy
```

Out[4]:

	Crop_Year	Area	Production	District_Name_24 PARAGANAS NORTH	District_Name_24 PARAGANAS SOUTH	District_Name_ADILABAD	District_Name_AGAR MALWA	District_Name_AGRA	District_Na
0	2000	1254.0	2000.0	0	0	0	0	0	
1	2000	2.0	1.0	0	0	0	0	0	
2	2000	102.0	321.0	0	0	0	0	0	
3	2000	176.0	641.0	0	0	0	0	0	



```
In [5]: from sklearn.model_selection import train_test_split
x = dummy.drop(["Production"], axis=1)
y = dummy["Production"]

# Splitting data set - 25% test dataset and 75%
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25, random_state=5)

print("x_train :",x_train.shape)
print("x_test :",x_test.shape)
print("y_train :",y_train.shape)
print("y_test :",y_test.shape)

x_train : (181770, 778)
x_test : (60591, 778)
y_train : (181770,)
y_test : (60591,)

In [6]: from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
model.fit(x_train,y_train)
preds = model.predict(x_test)
print("Random Forest Regressor")
print("Mean Squared Error: ",mean_squared_error(y_test,preds))
print("R2 score: ",r2_score(y_test,preds))

Random Forest Regressor
Mean Squared Error: 7205205429626.706
R2 score: 0.9752199327433567

In [7]: import joblib
joblib.dump(model,'Model.pkl')

Out[7]: ['Model.pkl']
```

Whenever the dataset gets updated, the model will be trained again and the pickle file will be updated at the latest.