In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn import metrics
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df=pd.read_csv('Crop_recommendation.csv')
df
```

Out[2]:

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice
...	...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

2200 rows × 8 columns

In [3]:
```
df.shape
```

Out[3]:
```
(2200, 8)
```

In [4]:
```
p = list(df.columns)
print(p)
#N = NITROGEN
#P = PHOSPHORUS
#K = POTASSIUM
#TEMPERATURE OF THE REGION
# HUMIDITY IN THAT REGION
# PH OF THE SOIL
# RAINFALL IN THAT REGION
# CROP THAT IS SUITABLE IN THAT REGION
```

```
['N', 'P', 'K', 'temperature', 'humidity', 'ph', 'rainfall', 'label']
```

In [5]:
```
(df.dtypes)
```

Out[5]:
```
N               int64
P               int64
K               int64
temperature   float64
humidity      float64
ph            float64
rainfall      float64
label          object
dtype: object
```

In [6]:
```
print(list(df['label'].unique()))
#LIST OF CROPS IN THE DATASET
```

```
['rice', 'maize', 'chickpea', 'kidneybeans', 'pigeonpeas', 'mothbeans', 'mungbe
an', 'blackgram', 'lentil', 'pomegranate', 'banana', 'mango', 'grapes', 'waterm
elon', 'muskmelon', 'apple', 'orange', 'papaya', 'coconut', 'cotton', 'jute',
'coffee']
```

In [7]:
```
df['label'].value_counts()
#22 DIFFERENT TYPES OF CROPS ARE PRESENT IN THE DATASET
#AND EACH CROP HAS 100 DIFFERENT SAMPLES
```

Out[7]:
```
maize          100
watermelon     100
banana         100
grapes         100
jute           100
apple          100
pigeonpeas     100
chickpea       100
cotton         100
mothbeans      100
orange         100
papaya         100
lentil         100
pomegranate    100
mungbean       100
muskmelon      100
mango          100
blackgram      100
coffee         100
coconut        100
rice           100
kidneybeans    100
Name: label, dtype: int64
```

In [8]:
```
1 df.describe()
```

Out[8]:

	N	P	K	temperature	humidity	ph	rainfall
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000
mean	50.551818	53.362727	48.149091	25.616244	71.481779	6.469480	103.463655
std	36.917334	32.985883	50.647931	5.063749	22.263812	0.773938	54.958389
min	0.000000	5.000000	5.000000	8.825675	14.258040	3.504752	20.211267
25%	21.000000	28.000000	20.000000	22.769375	60.261953	5.971693	64.551686
50%	37.000000	51.000000	32.000000	25.598693	80.473146	6.425045	94.867624
75%	84.250000	68.000000	49.000000	28.561654	89.948771	6.923643	124.267508
max	140.000000	145.000000	205.000000	43.675493	99.981876	9.935091	298.560117

In [9]:
```
1 df.head()
```

Out[9]:

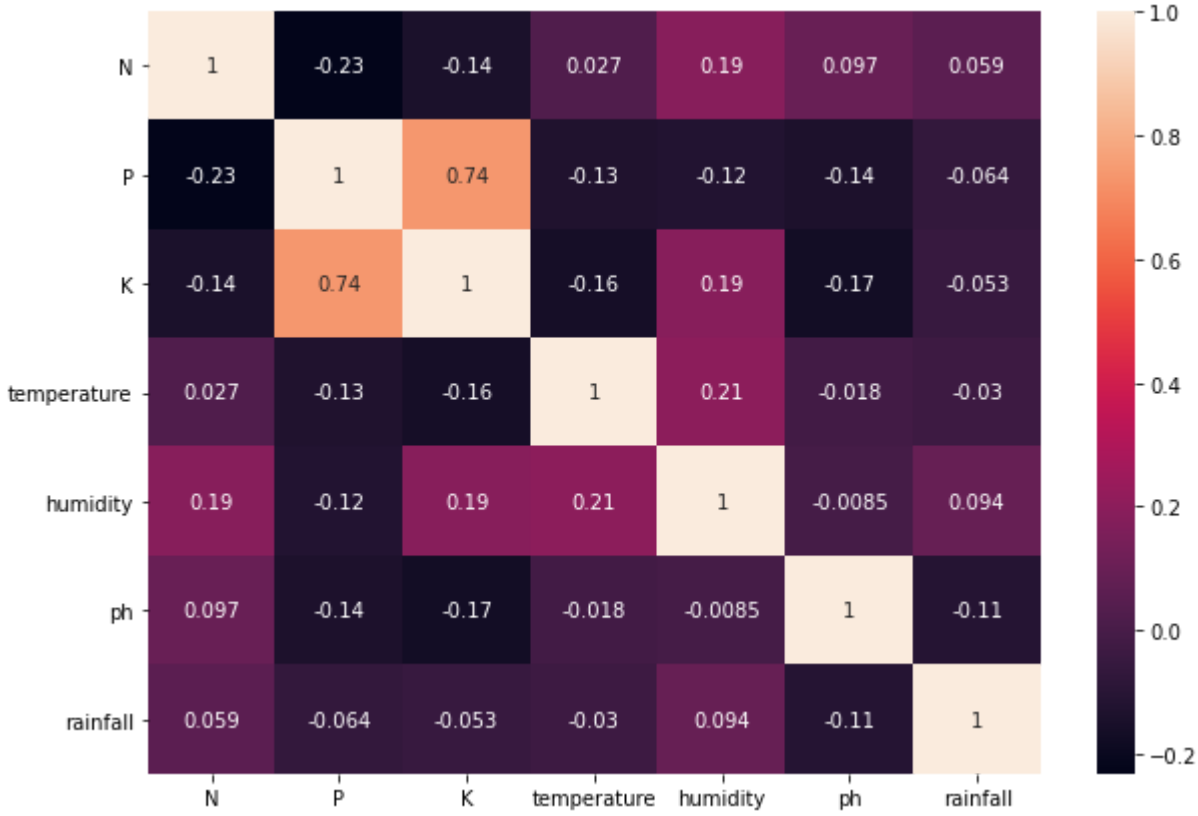
	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

In [10]:
```
1 df.tail()
```

Out[10]:

	N	P	K	temperature	humidity	ph	rainfall	label
2195	107	34	32	26.774637	66.413269	6.780064	177.774507	coffee
2196	99	15	27	27.417112	56.636362	6.086922	127.924610	coffee
2197	118	33	30	24.131797	67.225123	6.362608	173.322839	coffee
2198	117	32	34	26.272418	52.127394	6.758793	127.175293	coffee
2199	104	18	30	23.603016	60.396475	6.779833	140.937041	coffee

In [11]:
```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr(),annot = True)
plt.show()
#shows the correalation b/w to features in the dataset
#annot = writes the data value in each cell
```



In [12]:
```
features = df[['N','P','K','temperature','humidity','ph','rainfall']]
features
#THESE ARE THE FEATURES WHICH WE USE TO RECOMMEND A
#CROP IN A CERTAIN RANGE
```

Out[12]:

	N	P	K	temperature	humidity	ph	rainfall
0	90	42	43	20.879744	82.002744	6.502985	202.935536
1	85	58	41	21.770462	80.319644	7.038096	226.655537
2	60	55	44	23.004459	82.320763	7.840207	263.964248
3	74	35	40	26.491096	80.158363	6.980401	242.864034
4	78	42	42	20.130175	81.604873	7.628473	262.717340
...	...	...	...	...	...	...	...
2195	107	34	32	26.774637	66.413269	6.780064	177.774507
2196	99	15	27	27.417112	56.636362	6.086922	127.924610
2197	118	33	30	24.131797	67.225123	6.362608	173.322839
2198	117	32	34	26.272418	52.127394	6.758793	127.175293
2199	104	18	30	23.603016	60.396475	6.779833	140.937041

2200 rows × 7 columns

In [13]:
```
target = df[['label']]
target
#BY USING ABOVE FEATURES WE DECIDE
#WHICH CROP HAS TO BE GROWN AND HARVESTED IN THAT REGION
```

Out[13]:

	label
0	rice
1	rice
2	rice
3	rice
4	rice
...	...
2195	coffee
2196	coffee
2197	coffee
2198	coffee
2199	coffee

2200 rows × 1 columns

In [14]:
```
# Initialzing empty lists to append all
# model's name and corresponding accuracy score
models_accuracy = []
ml_models = []
```

In [15]:
```
from sklearn.model_selection import train_test_split
features_train,features_test,target_train,target_test = train_test_split(fea
                    test_size= 0.2,random_state = 1)
# 1)train_test_split splits the dataset into two parts for training
# and testing
# 2)test size = 0.2 means , 20% of dataset is used for testing
# and remaining 80% is used for training the model
# 3)random_state is basically used for reproducing your problem the
# same every time it is run.
```

In [16]:
```
features_train
#TRAINING DATASET
```

Out[16]:

	N	P	K	temperature	humidity	ph	rainfall
1863	1	12	30	27.754298	95.946438	5.562224	131.090008
987	11	10	45	22.630452	88.455772	6.397996	109.035760
375	5	74	21	16.244692	21.357939	5.591704	66.970533
1071	105	88	54	25.787498	84.511942	6.020445	114.200546
37	95	39	36	23.863305	83.152508	5.561399	285.249365
...	...	...	...	...	...	...	...
960	1	27	36	23.985988	93.342366	5.684995	104.991282
905	31	25	38	24.962732	92.405014	6.497367	109.416919
1096	92	81	52	28.010680	76.528081	5.891414	103.704078
235	28	58	81	17.475010	16.543148	6.180427	93.350343
1061	95	74	50	25.901131	80.471527	6.002482	110.103230

1760 rows × 7 columns

In [17]:
```
features_test
#TESTING DATASET
```

Out[17]:

	N	P	K	temperature	humidity	ph	rainfall
1276	25	129	195	17.986678	81.177121	5.777271	72.371277
1446	106	20	51	29.730197	90.970157	6.342573	20.490356
335	33	59	22	22.642369	21.593961	5.947000	122.388601
1458	89	9	47	29.471563	90.770696	6.668383	28.752261
2038	62	49	37	24.217446	82.852840	7.479248	166.136589
...	...	...	...	...	...	...	...
1508	22	144	196	21.911913	91.687481	6.499227	117.076128
1595	40	120	197	23.805938	92.488795	5.889481	119.633555
1032	105	74	45	25.145176	81.382041	6.098369	119.218154
1330	118	15	45	24.214957	84.205770	6.538006	48.011385
1263	37	135	205	11.827682	80.282719	5.510925	74.102251

440 rows × 7 columns

In [18]:
```
target_train
#LABEL OR OUTPUT FOR TRAINING DATA AS INPUT
```

Out[18]:

	label
1863	coconut
987	pomegranate
375	kidneybeans
1071	banana
37	rice
...	...
960	pomegranate
905	pomegranate
1096	banana
235	chickpea
1061	banana

1760 rows × 1 columns

In [19]:
```
target_test
#LABEL OR OUTPUT FOR TESTING DATA AS INPUT
```

Out[19]:

	label
1276	grapes
1446	muskmelon
335	kidneybeans
1458	muskmelon
2038	jute
...	...
1508	apple
1595	apple
1032	banana
1330	watermelon
1263	grapes

440 rows × 1 columns

MODELS

DECISION TREE MODEL

In [20]:
```
# from sklearn.tree import DecisionTreeClassifier
# DecisionTree = DecisionTreeClassifier(criterion = 'entropy',max_depth = 7,
# DecisionTree.fit(features_train,target_train)
# predicted = DecisionTree.predict(features_test)
# x = metrics.accuracy_score(target_test,predicted)
# acc.append(x)
# model.append('Decision Tree')
# print("Decision Tree's accuracy is", x * 100)

# print(classification_report(target_test,predicted))
# # entropy is measure of randomness
```

In [21]:
```
from sklearn import tree
```

In [22]:
```
model_dt = tree.DecisionTreeClassifier()
```

In [23]:
```
model_dt.fit(features_train ,target_train)
```

Out[23]:
```
DecisionTreeClassifier()
```

In [24]:
```
q = model_dt.score(features_test,target_test)
#GETTING THE ACCURACY VALUE OF THE MODEL
```

In [25]:
```
x = q*100
x
print("ACCURACY OF DECISION TREE MODEL IS: ",x)
```

```
ACCURACY OF DECISION TREE MODEL IS:  99.54545454545455
```

In [26]:
```
models_accuracy.append(x)
ml_models.append('DECISION_TREE')
```

In [27]:
```
print(list(model_dt.predict([[70,42,63,30.879744,82.002744,16.502985,45.9355
```

```
['banana']
```

In [28]:
```
def predict_dt_crop():
    lst=list(range(7))
    lst[0]=int(input("PLEASE ENTER VALUE OF NITROGEN: "))
    lst[1]=int(input("PLEASE ENTER VALUE OF PHOSPHORUS: "))
    lst[2]=int(input("PLEASE ENTER VALUE OF POTASSIUM: "))
    lst[3]=float(input("PLEASE ENTER VALUE OF TEMPERATURE: "))
    lst[4]=float(input("PLEASE ENTER VALUE OF HUMIDITY: "))
    lst[5]=float(input("PLEASE ENTER VALUE OF PH: "))
    lst[6]=float(input("PLEASE ENTER AMOUNT OF RAINFALL: "))
    print("THE PREFERED CROP IN YOUR REGION IS: ",
          model_dt.predict([lst]))
    print("👍 👍HAPPY FARMING😁 😁")
```

In [29]:
```
predict_dt_crop()
```

```
PLEASE ENTER VALUE OF NITROGEN: 3
PLEASE ENTER VALUE OF PHOSPHORUS: 49
PLEASE ENTER VALUE OF POTASSIUM: 27
PLEASE ENTER VALUE OF TEMPERATURE: 64
PLEASE ENTER VALUE OF HUMIDITY: 3.69
PLEASE ENTER VALUE OF PH: 3
PLEASE ENTER AMOUNT OF RAINFALL: 32
THE PREFERED CROP IN YOUR REGION IS:  ['kidneybeans']
👍 👍HAPPY FARMING😁 😁
```

NAIVE BAYES

In [30]:
```
from sklearn.naive_bayes import GaussianNB
model_nb = GaussianNB()
```

In [31]:
```
model_nb.fit(features_train,target_train)
```

Out[31]:
```
GaussianNB()
```

In [32]:
```
y1 = model_nb.score(features_test,target_test)
y = y1*100
print("ACCURACY OF NAIVE BAYES MODEL IS: ",y)
```

```
ACCURACY OF NAIVE BAYES MODEL IS:  99.54545454545455
```

In [33]:
```
target_test[:10]
```

Out[33]:

	label
1276	grapes
1446	muskmelon
335	kidneybeans
1458	muskmelon
2038	jute
1314	watermelon
389	kidneybeans
1639	orange
2004	jute
403	pigeonpeas

In [34]:
```
print(list(model_nb.predict(features_test[:10])))
```

```
['grapes', 'muskmelon', 'kidneybeans', 'muskmelon', 'jute', 'watermelon', 'kidn
eybeans', 'orange', 'jute', 'pigeonpeas']
```

In [35]:
```
def predict_nb_crop():
    lst=list(range(7))
    lst[0]=int(input("PLEASE ENTER VALUE OF NITROGEN: "))
    lst[1]=int(input("PLEASE ENTER VALUE OF PHOSPHORUS: "))
    lst[2]=int(input("PLEASE ENTER VALUE OF POTASSIUM: "))
    lst[3]=float(input("PLEASE ENTER VALUE OF TEMPERATURE: "))
    lst[4]=float(input("PLEASE ENTER VALUE OF HUMIDITY: "))
    lst[5]=float(input("PLEASE ENTER VALUE OF PH: "))
    lst[6]=float(input("PLEASE ENTER AMOUNT OF RAINFALL: "))
    print("THE PREFERED CROP IN YOUR REGION IS: ",
          model_nb.predict([lst]))
    print("👍 👍HAPPY FARMING😁😁")
```

In [36]:
```
predict_nb_crop()
```

```
PLEASE ENTER VALUE OF NITROGEN: 3
PLEASE ENTER VALUE OF PHOSPHORUS: 49
PLEASE ENTER VALUE OF POTASSIUM: 18
PLEASE ENTER VALUE OF TEMPERATURE: 27
PLEASE ENTER VALUE OF HUMIDITY: 64
PLEASE ENTER VALUE OF PH: 3
PLEASE ENTER AMOUNT OF RAINFALL: 32
THE PREFERED CROP IN YOUR REGION IS:  ['mothbeans']
👍👍HAPPY FARMING😁😁
```

In [37]:
```
models_accuracy.append(y)
ml_models.append('NAIVE_BAYES')
```

SUPPORT VECTOR MACHINE

In [38]:
```
from sklearn.svm import SVC
```

In [39]:
```
model_svm = SVC()
```

In [40]:
```
model_svm.fit(features_train,target_train)
```

Out[40]:
```
SVC()
```

In [41]:
```
z1 = model_svm.score(features_test,target_test)
z = z1*100
print("ACCURACY OF SUPPORT VECTOR MACHINE MODEL IS:",z)
```

```
ACCURACY OF SUPPORT VECTOR MACHINE MODEL IS: 97.95454545454545
```

In [42]:
```
1 target_test[:10]
```

Out[42]:

	label
1276	grapes
1446	muskmelon
335	kidneybeans
1458	muskmelon
2038	jute
1314	watermelon
389	kidneybeans
1639	orange
2004	jute
403	pigeonpeas

In [43]:
```
1 print(list(model_svm.predict(features_test[:10])))
```

```
['grapes', 'muskmelon', 'kidneybeans', 'muskmelon', 'jute', 'watermelon', 'kidn
eybeans', 'orange', 'jute', 'pigeonpeas']
```

In [44]:
```
1 models_accuracy.append(z)
2 ml_models.append('SVM')
```

In [45]:
```
1  def predict_svm_crop():
2      lst=list(range(7))
3      lst[0]=int(input("PLEASE ENTER VALUE OF NITROGEN: "))
4      lst[1]=int(input("PLEASE ENTER VALUE OF PHOSPHORUS: "))
5      lst[2]=int(input("PLEASE ENTER VALUE OF POTASSIUM: "))
6      lst[3]=float(input("PLEASE ENTER VALUE OF TEMPERATURE: "))
7      lst[4]=float(input("PLEASE ENTER VALUE OF HUMIDITY: "))
8      lst[5]=float(input("PLEASE ENTER VALUE OF PH: "))
9      lst[6]=float(input("PLEASE ENTER AMOUNT OF RAINFALL: "))
10     print("THE PREFERED CROP IN YOUR REGION IS: ",
11           model_svm.predict([lst]))
12     print("👍 👍HAPPY FARMING😁😁")
```

In [46]:
```
1 predict_nb_crop()
```

```
PLEASE ENTER VALUE OF NITROGEN: 3
PLEASE ENTER VALUE OF PHOSPHORUS: 49
PLEASE ENTER VALUE OF POTASSIUM: 18
PLEASE ENTER VALUE OF TEMPERATURE: 27
PLEASE ENTER VALUE OF HUMIDITY: 64
PLEASE ENTER VALUE OF PH: 3
PLEASE ENTER AMOUNT OF RAINFALL: 32
THE PREFERED CROP IN YOUR REGION IS:  ['mothbeans']
👍 👍HAPPY FARMING😁😁
```

RANDOM FOREST

In [47]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [48]:
```
model_rf = RandomForestClassifier()
```

In [49]:
```
model_rf.fit(features_train,target_train)
```

Out[49]:
```
RandomForestClassifier()
```

In [50]:
```
a1 = model_rf.score(features_test,target_test)
a = a1*100
print("ACCURACY OF RANDOM FOREST MODEL IS: ",a)
```

```
ACCURACY OF RANDOM FOREST MODEL IS:  99.77272727272727
```

In [51]:
```
target_test[:5]
```

Out[51]:

	label
1276	grapes
1446	muskmelon
335	kidneybeans
1458	muskmelon
2038	jute

In [52]:
```
print(list(model_rf.predict(features_test[:5])))
```

```
['grapes', 'muskmelon', 'kidneybeans', 'muskmelon', 'jute']
```

In [53]:
```
models_accuracy.append(a)
ml_models.append('RANDOM_FOREST')
```

In [54]:
```
def predict_rf_crop():
    lst=list(range(7))
    lst[0]=int(input("PLEASE ENTER VALUE OF NITROGEN: "))
    lst[1]=int(input("PLEASE ENTER VALUE OF PHOSPHORUS: "))
    lst[2]=int(input("PLEASE ENTER VALUE OF POTASSIUM: "))
    lst[3]=float(input("PLEASE ENTER VALUE OF TEMPERATURE: "))
    lst[4]=float(input("PLEASE ENTER VALUE OF HUMIDITY: "))
    lst[5]=float(input("PLEASE ENTER VALUE OF PH: "))
    lst[6]=float(input("PLEASE ENTER AMOUNT OF RAINFALL: "))
    print("THE PREFERED CROP IN YOUR REGION IS: ",
          model_rf.predict([lst]))
    print("👍 👍HAPPY FARMING😁 😁")
```

In [55]:
```
predict_rf_crop()
```

```
PLEASE ENTER VALUE OF NITROGEN: 3
PLEASE ENTER VALUE OF PHOSPHORUS: 49
PLEASE ENTER VALUE OF POTASSIUM: 18
PLEASE ENTER VALUE OF TEMPERATURE: 27
PLEASE ENTER VALUE OF HUMIDITY: 64
PLEASE ENTER VALUE OF PH: 3
PLEASE ENTER AMOUNT OF RAINFALL: 32
THE PREFERED CROP IN YOUR REGION IS:  ['mothbeans']
👍👍HAPPY FARMING😁😁
```

XGBOOST

In [56]:
```
import xgboost as xgb
```

In [57]:
```
model_xb = xgb.XGBClassifier()
```

In [58]:
```
model_xb.fit(features_train,target_train)
b1  = model_xb.score(features_test,target_test)
```

```
[19:41:23] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.
1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric
used with the objective 'multi:softprob' was changed from 'merror' to 'mloglos
s'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

In [59]:
```
b = b1*100
print("ACCURACY OF XGBOOST MODEL IS: ",b)
#WE GOT ACCURACY SAME AS DECISION TREE AND NAIVE BAYES
```

```
ACCURACY OF XGBOOST MODEL IS:  99.54545454545455
```

In [60]:
```
target_test[5:10]
```

Out[60]:

	label
1314	watermelon
389	kidneybeans
1639	orange
2004	jute
403	pigeonpeas

In [61]:
```
print(list(model_xb.predict(features_test[5:10])))
```

```
['watermelon', 'kidneybeans', 'orange', 'jute', 'pigeonpeas']
```

In [62]:
```
models_accuracy.append(b)
ml_models.append('XG_BOOST')
```

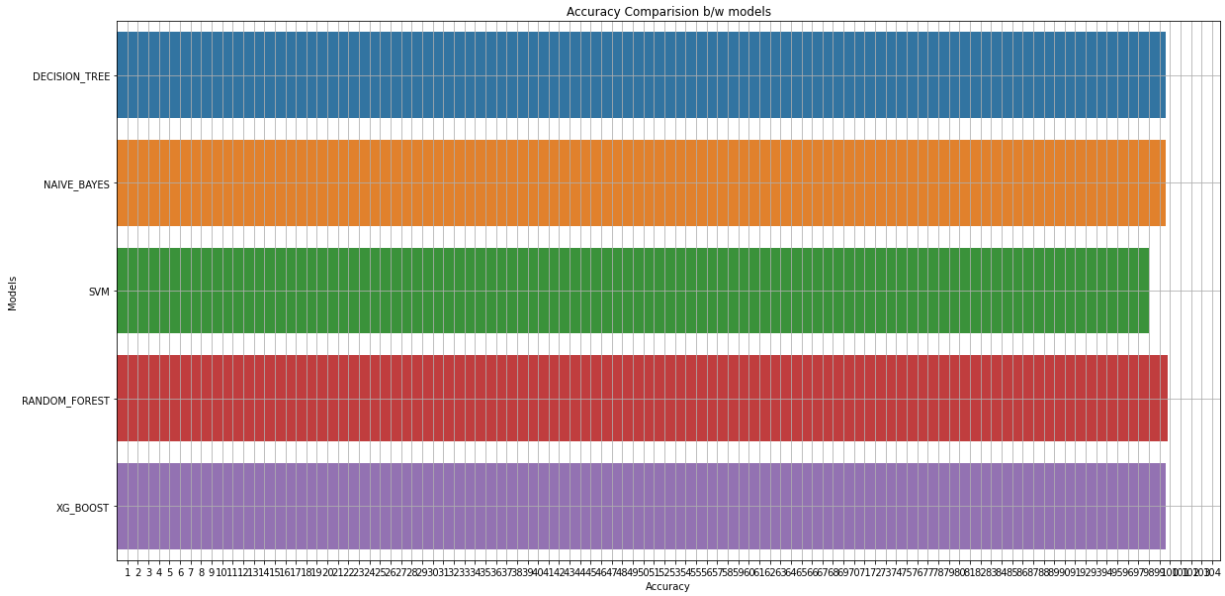
In [63]:
```
ml_models
```

Out[63]:
```
['DECISION_TREE', 'NAIVE_BAYES', 'SVM', 'RANDOM_FOREST', 'XG_BOOST']
```

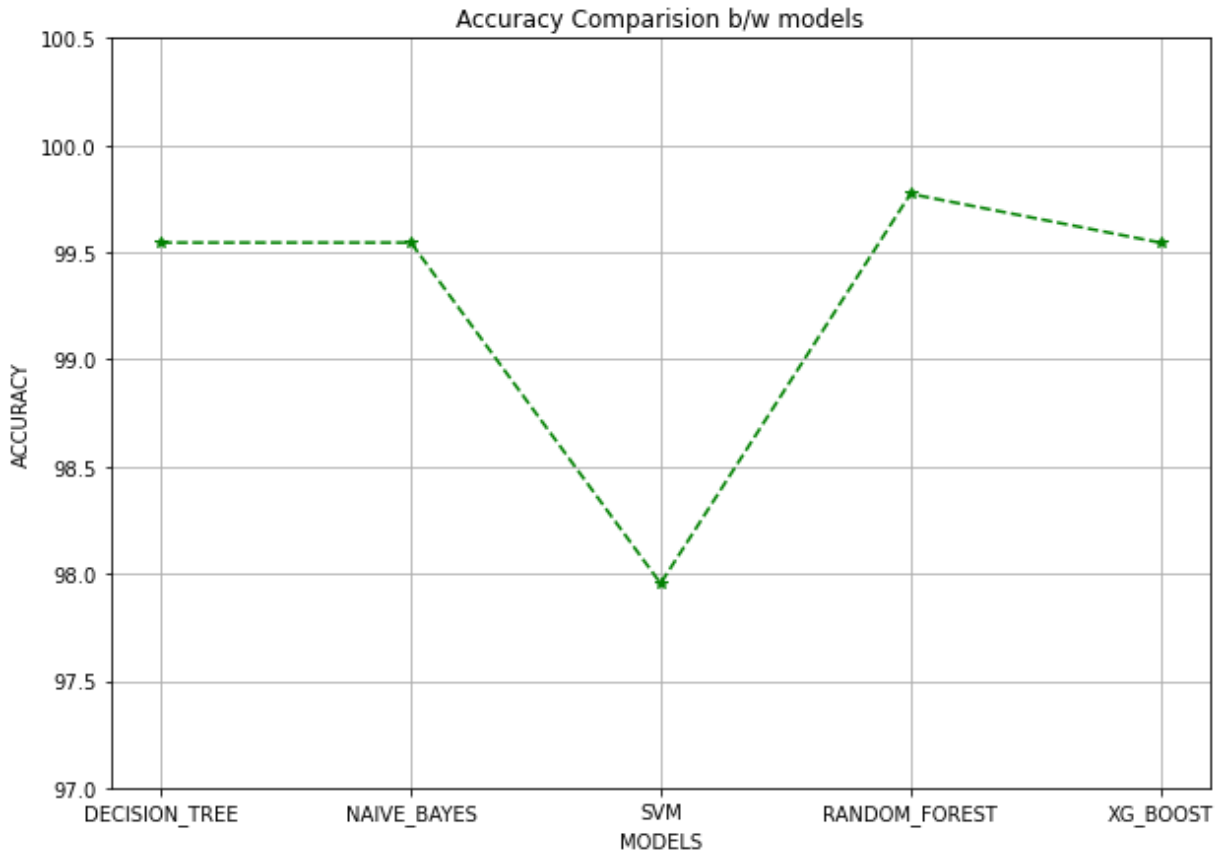
In [65]:
```
print(models_accuracy)
```

```
[99.54545454545455, 99.54545454545455, 97.95454545454545, 99.77272727272727, 9
9.54545454545455]
```

In [90]:
```
plt.figure(figsize = [20,10])
plt.title('Accuracy Comparision b/w models')
plt.xlabel('Accuracy')
plt.ylabel('Models')
sns.barplot(x = models_accuracy,y = ml_models)
plt.grid()
plt.xticks(np.arange(1,105,1))
plt.show()
```



In [92]:
```
plt.figure(figsize = [10,7])
plt.title('Accuracy Comparision b/w models')
plt.plot(models_accuracy,ls='--',marker='*',color='g')
xlabels=ml_models
plt.xticks(range(len(xlabels)),xlabels)
plt.yticks(np.arange(97,101,0.5))
plt.xlabel('MODELS')
plt.ylabel('ACCURACY')
plt.grid()
plt.show()
```



In [96]:
```
accuracy_of_models = dict(zip(ml_models,models_accuracy))
for i,j in accuracy_of_models.items():
    print(i,'=====>',j,'%')
```

```
DECISION_TREE =====> 99.54545454545455 %
NAIVE_BAYES =====> 99.54545454545455 %
SVM =====> 97.95454545454545 %
RANDOM_FOREST =====> 99.77272727272727 %
XG_BOOST =====> 99.54545454545455 %
```

FINAL PREDICTION

In [97]:
```
predict_rf_crop()
```

```
PLEASE ENTER VALUE OF NITROGEN: 90
PLEASE ENTER VALUE OF PHOSPHORUS: 42
PLEASE ENTER VALUE OF POTASSIUM: 43
PLEASE ENTER VALUE OF TEMPERATURE: 23.603
PLEASE ENTER VALUE OF HUMIDITY: 60.3
PLEASE ENTER VALUE OF PH: 6.7
PLEASE ENTER AMOUNT OF RAINFALL: 140.91
THE PREFERED CROP IN YOUR REGION IS:  ['coffee']
👍👍HAPPY FARMING😁😁
```

In [ ]:
```

```