#CANDIDATE

```
import numpy as np
import pandas as pd
data =
pd.read_csv("ENJOYSPORT.csv")
concepts = np.array(data.iloc[:, :-1])
target = np.array(data.iloc[:, -1])
def learn(concepts, target):
    s_h = concepts[0].copy()
    g_h = [["?" for _ in
range(len(s_h))] for _ in
range(len(s_h))]

    for i, h in enumerate(concepts):
        if target[i] == "yes":
            for x in range(len(s_h)):
                if h[x] != s_h[x]:
                    s_h[x] = '?'
                    g_h[x][x] = '?'
        else:
            for x in range(len(s_h)):
                if h[x] != s_h[x]:
                    g_h[x][x] = s_h[x]
                else:
                    g_h[x][x] = '?'

    g_h = [g for g in g_h if g != ['?' for
_ in range(len(s_h))]]
    return s_h, g_h
s_final, g_final = learn(concepts,
target)
print("Final Specific_h:\n", s_final)
print("Final General_h:\n", g_final)
```

#NAIVE-BAYES:

```
import numpy as np
import pandas as pd
data =
pd.read_csv("play_tennis.csv")
data.drop(["day"], axis=1,
inplace=True)
def nb_predict(data, target,
test_inst):
    target_counts =
data[target].value_counts().to_dict(
)
    target_probs =
(data[target].value_counts() /
data.shape[0]).to_dict()
    pred_probs = {}

    for tgt, count in
target_counts.items():
        attr_prob = 1
        data_subset =
data[data[target] == tgt]

        for attr, val in test_inst.items():
            attr_prob *=
data_subset[data_subset[attr] ==
val].shape[0] /
data_subset.shape[0]

        pred_probs[tgt] =
target_probs[tgt] * attr_prob

    return pred_probs
test_inst = {"outlook": "Sunny",
"temp": "Cool", "humidity": "High",
"wind": "Strong"}
pred = nb_predict(data.copy(),
data.columns[-1], test_inst)
print("Prediction is:", pred)
```

#ID3:

```
import pandas as pd
import numpy as np
from sklearn.datasets import
load_iris
from sklearn import tree
iris=load_iris()
print(iris.feature_names)
print(iris.target_names)
removed =[0,50,100]
new_target =
np.delete(iris.target,removed)
new_data =
np.delete(iris.data,removed,
axis=0)
#train classifier
clf = tree.DecisionTreeClassifier()
# defining decision tree classifier
clf=clf.fit(new_data,new_target)
# train data on new data and
new target
prediction =
clf.predict(iris.data[removed]) #
assign removed data as input
print("Original
Labels",iris.target[removed])
print("Labels
Predicted",prediction)
tree.plot_tree(clf)
```

#NAIVE_BAYSE_B

```
import numpy as np
from sklearn.datasets import
fetch_20newsgroups
from sklearn.feature_extraction.text
import TfidfVectorizer
from sklearn.model_selection
import train_test_split
from sklearn.naive_bayes import
MultinomialNB
from sklearn.metrics import
accuracy_score, precision_score,
recall_score, classification_report

newsgroups =
fetch_20newsgroups(subset='all')
X, y = newsgroups.data,
newsgroups.target

vectorizer =
TfidfVectorizer(stop_words='english
')
X_tfidf = vectorizer.fit_transform(X)

X_train, X_test, y_train, y_test =
train_test_split(X_tfidf, y,
test_size=0.2, random_state=42)

model = MultinomialNB()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
accuracy = accuracy_score(y_test,
predictions)
precision = precision_score(y_test,
predictions, average='macro')
recall = recall_score(y_test,
predictions, average='macro')
classification_rep =
classification_report(y_test,
predictions,
target_names=newsgroups.target_
names)
```

```
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print('\nClassification Report:\n',
classification_rep)

#SVM_WINEDATASET
import numpy as np
from sklearn import svm
from sklearn.metrics import
accuracy_score
from sklearn.preprocessing import
StandardScaler
from sklearn.datasets import
load_wine
from sklearn.model_selection
import train_test_split
import matplotlib.pyplot as plt

wine = load_wine()
data, target = wine.data,
wine.target
print(wine)

train_data, test_data, train_target,
test_target = train_test_split(data,
target, test_size=0.2,
random_state=42)

scaler = StandardScaler()
train_data =
scaler.fit_transform(train_data)
test_data =
scaler.transform(test_data)

model = svm.SVC(kernel='linear')
model.fit(train_data, train_target)

predictions =
model.predict(test_data)
accuracy =
accuracy_score(test_target,
predictions)
print(f'Test accuracy: {accuracy}')

sample_index = 0
new_sample =
test_data[sample_index].reshape(1,
-1)
predicted_label =
model.predict(new_sample)
actual_label =
test_target[sample_index]

print(f'Predicted label:
{predicted_label[0]}, Actual label:
{actual_label}')

feature_names =
wine.feature_names
target_names = wine.target_names

plt.figure(figsize=(10, 6))
plt.barh(feature_names,
test_data[sample_index])
plt.title(f'Predicted:
{target_names[predicted_label[0]]}
, Actual:
{target_names[actual_label]}')
plt.xlabel('Scaled Feature Values')
plt.show()
```