Name of Experiment.....
Experiment No.....01.....
Date.....30/01/2022.....
Experiment Result.....
Page No.

FIND-S algorithum for finding the maximally Specific hypothesis Consistent with the training Example.

```
import csv
num_attributes = 6
a = []
with open('Enjoy Sport. csv', 'r') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        a.append(row)
print("\n Given Training Data: \n")
print(a)
hypothesis = ['0'] * num_attributes
print("\n The initial hypothesis: ", hypothesis)
for j in range(0, num_attributes):
    hypothesis[j] = a[0][j]
    print(hypothesis)
Print("\n Find S: Finding a maximally Specific Hypothesis:\n")
for i in range(0, len(a)):
    if a[i][num_attributes] == 'yes':
        for j in range(0, num_attributes):
            if a[i][j] != hypothesis[j]:
                hypothesis[j] = '?'
print("For Training instance No. ", i, ", The maximally
    Specific Hypothesis is ", hypothesis)
print("\n The maximally Specific Hypothesis for the given Set of
    training Example is: \n")
Print(hypothesis)
```

01/12/22

Name of Experiment.....
Experiment No.....02.....
Date.....6/2/2023.....
Experiment Result.....
Page No.

Implement and demonstrate the working of the decision tree based ID3 algorithm. use an appropriate data set for building the decision tree, find the accuracy and apply this knowledge to classify a new sample. Read the training data from a .csv file, Python ML library classes can be used for this Problem. Compute the various Performance metrics classifier.

```
from Pandas import read_csv
from sklearn.model_selection import KFold
df = read_csv('iris.csv')
array = df.values
x = array[:,0:-1]
y = array[:,-1]
kfold = KFold(n_splits=10, shuffle=True, random_state=0)

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
model = DecisionTreeClassifier(criterion="entropy", random_state=0)
print("(Accuracy = {:.2f}%, SD = {:.2f})".format(result.mean()
        *100, result.std()))
model = model.fit(x,y)
y_pred = model.predict([[5.1, 3.7, 1.5, 0.4]])

from sklearn.model_selection import cross_val_predict
from sklearn.metrics import Confusion_matrix
from sklearn.metrics import classification_report
y_pred = Confusion_matrix(y, y_pred)
Print('\n', conf_mat)
report = Classification_report(y, y_pred)
print('\n', report)
```

Name of Experiment..... Date..... Page No.

Experiment No..... Experiment Result.....

```
from sklearn import tree
import graphviz
fn = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
      'petal width (cm)']
cn = ['0', '1', '2']

tree.export_graphviz(model, out_file = "tree.dot",
                     feature_names = fn, class_name = cn, filled = True)
```

6/3/22

Name of Experiment..... Date.....13/2/2023..... Page No.

Experiment No.....03..... Experiment Result.....

3. Implement and demonstrate the working of an ensemble technique based on decision tree. Use an appropriate data set. Read the training data from a .csv file. Python ML library classes can be used for this problem. Compute the various performance metrics of the classifier

```
from pandas import read_csv
from sklearn model_selection import KFold
df = read_csv ('iris.csv')
array = df.values
x = array [:, 0:-1]
y = array [:, -1]
kfold = KFold (n_splits = 10, shuffle = True, random_state = 0)

from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import cross_val_score
model = BaggingClassifier (n_estimators = 20, random_state = 0)
model = RandomForestClassifier (random_state = 0, n_estimates = 100,
        criterion = "entropy")
model = ExtraTreeClassifier (n_estimators = 100, random_state = 0)
result = cross_val_score (model, x, y, cv = kfold, scoring = 'accuracy')
print ("( Accuracy = %.2f%%, 3D= %.2f)", format (result.mean()
                * 100. result.std())

model = model.fit (x, y)
y_pred = model.predict ([[5.1, 3.7, 1.5, 0.4]])
Print (y_pred)
```

Name of Experiment..... Date..... Page No.

Experiment No..... Experiment Result.....

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
y_pred = cross_val_predict(model, x, y, cv=kfold)
conf_mat = confusion_matrix(x, y_pred)
print('\n', conf_mat)
report = classification_report(x, y_pred)
print('\n', report)

import graphviz
from sklearn.tree import export_graphviz
fn = ['sepal_length(cm)', 'sepal_width(cm)', 'Petal_length(cm)', 'petal_length
      (cm)']
cn = ['0', '1', '2']
dot_data = export_graphviz(model.estimators_[99],
                           out_file = 'tree.dot',
                           feature_names = fn,
                           class_names = cn,
                           filled = True, impurity = True)
```

6/12/23

Name of Experiment..... Date..... Page No.

Experiment No.....4..... Experiment Result.....

4. Demonstrate the working of an Artificial Neural Network by implementing the Propagation algorithm. use an appropriate data set. Read the training data from a .csv file Python ML library classes in the used for this problem, Compute the various performance metrics of the classifier.

```
from pandas.impport read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.neutral.network import MLPclassifier

df = read_csv('Raisins.csv')
array = df.values
X = array[:,0:-1]
y = array[:,-1]
kfold = kFold(n_splits = 10, shuffle = True, random_state = 0)
model = MLPclassifier(hidden_layer_sizes = (50), shuffle = True, random_state
                      = 0, max_iter = 100)
result = cross_val_score(model, x, y, cv = kfold, scoring = 'accuracy')
print("Accuracy = %.2f %%"%(result.mean()*100))
y_pred = cross_val_predict(model, x, y, cv = kfold)
conf_mat = confusion_matrix(y, y_pred)
print('\n', conf_mat)
```