find s

```
import csv
import time

start = time.time()   # Start time

data = []

# Read CSV file
with open('trainingdata.csv','r') as file:
   reader = csv.reader(file)
   next(reader)

   for row in reader:
      data.append(row)

# Initialize hypothesis
h = data[0][:-1]

print("Initial Hypothesis:",h)

# Apply FIND-S
for row in data:
   if row[-1] == "Yes":
      for i in range(len(h)):
         if h[i] != row[i]:
            h[i] = '?'
```

```
print("Final Hypothesis:",h)


end = time.time()   # End time


print("Execution Time:", end - start,"seconds")
```

find s using file handling

```
import csv


#using file handling

# open file

f = open("trainingdata.csv","r")


data = list(csv.reader(f))


# remove header

data = data[1:]


# initialize hypothesis

h = data[0][:-1]


print("Initial Hypothesis:", h)


# FIND-S algorithm

for row in data:

   if row[-1] == "Yes":

      for i in range(len(h)):
```

```
        if h[i] != row[i]:
            h[i] = '?'

print("Final Hypothesis:", h)

f.close()
```

exp 2 candidate elimination

```
import csv

data = []

# Read CSV file
with open('trainingdata.csv','r') as file:
   reader = csv.reader(file)
   next(reader)
   for row in reader:
      data.append(row)

# Number of attributes
n = len(data[0]) - 1

# Initialize S and G
S = data[0][:-1]
G = [["?" for i in range(n)]]

print("Initial S:", S)
print("Initial G:", G)
```

```
for row in data:
    if row[-1] == "Yes":  # Positive example
        for i in range(n):
            if S[i] != row[i]:
                S[i] = "?"

    else:  # Negative example
        for i in range(n):
            if S[i] != row[i]:
                G.append(S.copy())

print("\nFinal Specific Hypothesis (S):", S)
print("Final General Hypothesis (G):", G)
```

exp 3 decision tree id3

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier

# Read dataset
data = pd.read_csv("tennis.csv")

# Convert categorical values to numbers
le = LabelEncoder()

for column in data.columns:
    data[column] = le.fit_transform(data[column])
```

```
# Split features and target
X = data.iloc[:,:-1]
y = data.iloc[:,-1]

# Create Decision Tree using entropy (ID3)
model = DecisionTreeClassifier(criterion='entropy')

# Train model
model.fit(X,y)

# New sample (example)
sample = [[2,1,0,1]]

# Predict result
prediction = model.predict(sample)

print("Prediction:", prediction)
```

perform and implement knn and decision tree classifier on the iris datasets

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load iris dataset
```

```
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=42)

# KNN Classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,y_train)
y_pred1 = knn.predict(X_test)

# Decision Tree Classifier
dt = DecisionTreeClassifier()
dt.fit(X_train,y_train)
y_pred2 = dt.predict(X_test)

# Accuracy
print("KNN Accuracy:",accuracy_score(y_test,y_pred1))
print("Decision Tree Accuracy:",accuracy_score(y_test,y_pred2))
```

```
Sky,AirTemp,Humidity,Wind,Water,Forecast,EnjoySport
Sunny,Warm,Normal,Strong,Warm,Same,Yes
Sunny,Warm,High,Strong,Warm,Same,Yes
Rainy,Cold,High,Strong,Warm,Change,No
Sunny,Warm,High,Strong,Cool,Change,Yes
```

Outlook,Temperature,Humidity,Wind,PlayTennis

Sunny,Hot,High,Weak,No

Sunny,Hot,High,Strong,No

Overcast,Hot,High,Weak,Yes

Rain,Mild,High,Weak,Yes

Rain,Cool,Normal,Weak,Yes

Rain,Cool,Normal,Strong,No

Overcast,Cool,Normal,Strong,Yes

Sunny,Mild,High,Weak,No

Sunny,Cool,Normal,Weak,Yes

Rain,Mild,Normal,Weak,Yes

Sunny,Mild,Normal,Strong,Yes

Overcast,Mild,High,Strong,Yes

Overcast,Hot,Normal,Weak,Yes

Rain,Mild,High,Strong,No

EXP 4

```
# Import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# Load dataset
iris = load_iris()
```

```
# Input features and target
X = iris.data
y = iris.target

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.3, random_state=42
)

knn = KNeighborsClassifier(n_neighbors=3)

knn.fit(X_train, y_train)

y_pred_knn = knn.predict(X_test)

print("KNN Accuracy:", accuracy_score(y_test, y_pred_knn)*100, "%")

print("KNN Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_knn))


dt = DecisionTreeClassifier()

dt.fit(X_train, y_train)

y_pred_dt = dt.predict(X_test)

print("\nDecision Tree Accuracy:", accuracy_score(y_test, y_pred_dt)*100, "%")

print("Decision Tree Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_dt))


# Plot Decision Tree
plt.figure(figsize=(10,6))
plot_tree(dt, feature_names=iris.feature_names,
      class_names=iris.target_names,
      filled=True)

plt.show()
```