**Experiment – 1**

**Aim:** Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

**Program:**

import pandas as pd

import numpy as np

data = pd.read\_csv("enjoysport.csv")

print(data)

d = np.array(data.iloc[:, 0:-1])

print("The attributes are: ", d)

t = np.array(data.iloc[:, -1])

print("The target is: ", t)

def train(c, t):

for i, val in enumerate(t):

if val == 'yes':

specific\_h = c[i].copy()

break

for i, val in enumerate(c):

if t[i] == 'yes':

for j in range(len(specific\_h)):

if val[j] != specific\_h[j]:

specific\_h[j] = '?'

else:

pass

return specific\_h

print("The Final hypothesis is : ",train(d, t))

**Output:** Sky AirTemp Humidity Wind Water Forecast EnjoySport

0 Sunny Warm Normal Strong Warm Same yes

1 Sunny Warm High Strong Warm Same yes

2 Rainy Cold High Strong Warm Change no

3 Sunny Warm High Strong Cool Change yes

The attributes are: [['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']

['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']

['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]

The target is: ['yes' 'yes' 'no' 'yes']

The Final hypothesis is : ['Sunny' 'Warm' '?' 'Strong' '?' '?']

**Experiment – 2**

**Aim: F**or a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate - Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples

**Program:**

import numpy as np

import pandas as pd

data = pd.read\_csv("enjoysport.csv")

concepts = np.array(data.iloc[:, 0:-1])

print("Instances are : ", concepts)

target = np.array(data.iloc[:, -1])

print("Target values are : ", target)

def learn(concepts, target):

specific\_h = concepts[0].copy()

print("initialization of specific\_h and general\_h")

print("Specific Boundary: ", specific\_h)

general\_h = [["?" for i in range(len(specific\_h))]

for i in range(len(specific\_h))]

print("Generic Boundary: ", general\_h)

for i, h in enumerate(concepts):

if target[i] == "yes":

print("Instance is Positive ")

for x in range(len(specific\_h)):

if h[x] != specific\_h[x]:

specific\_h[x] = '?'

general\_h[x][x] = '?'

if target[i] == "no":

print("Instance is Negative ")

for x in range(len(specific\_h)):

if h[x] != specific\_h[x]:

general\_h[x][x] = specific\_h[x]

else:

general\_h[x][x] = '?'

indices = [i for i, val in enumerate(general\_h) if val == [

'?', '?', '?', '?', '?', '?']]

for i in indices:

general\_h.remove(['?', '?', '?', '?', '?', '?'])

return specific\_h, general\_h

s\_final, g\_final = learn(concepts, target)

print("Final Specific\_h:", s\_final, sep="\n")

print("Final General\_h:", g\_final, sep="\n")

**Output:**

Instances are :

[['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']

['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']

['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]

Target values are : ['yes' 'yes' 'no' 'yes']

initialization of specific\_h and general\_h

Specific Boundary: ['Sunny' 'Warm' 'Normal' 'Strong' 'Warm' 'Same']

Generic Boundary: [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance is Positive

Instance is Positive

Instance is Negative

Instance is Positive

Final Specific\_h:

['Sunny' 'Warm' '?' 'Strong' '?' '?']

Final General\_h:

[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]

**Experiment – 3**

**Aim:** Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

**Program:**

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import numpy as np

from sklearn import tree

data = load\_iris()

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

x\_train, x\_test, y\_train, y\_test = train\_test\_split(df[data.feature\_names], df['target'], random\_state=0)

dt = DecisionTreeClassifier(max\_depth=2, random\_state=0)

dt.fit(x\_train, y\_train)

dt.predict(x\_test)

tree.plot\_tree(dt)

fn = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

cn = ['setosa', 'versicolor', 'virginica']

fig,axes = plt.subplots(nrows=1, ncols=1, figsize=(4, 4), dpi=300)

tree.plot\_tree(dt, feature\_names=fn, class\_names=cn, filled=True)

fig.savefig('imagename.png')

y\_pred = dt.predict(x\_test)

print(y\_pred)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

**Output:**

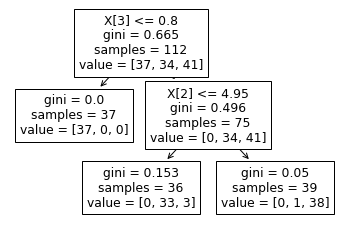
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 1 1 0 0 1 0 0 1 1 0 2 1 0 1 2 1 0

2]

[[13 0 0]

[ 0 15 1]

[ 0 3 6]]



**Experiment – 4**

**Aim:** Exercises to solve the real-world problems using the following machine learning methods: a) Linear Regression b) Logistic Regression c) Binary Classifier

**a) Linear Regression**

**Program:**

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

data\_set = pd.read\_csv("salary\_data.csv")

x = data\_set.iloc[:, :-1].values

y = data\_set.iloc[:, 1].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=1/3, random\_state=0)

model = LinearRegression()

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_test)

x\_pred = model.predict(x\_train)

plt.scatter(x\_train, y\_train, color="green")

plt.plot(x\_train, x\_pred, color="red")

plt.title("Salary vs Experience(Training Dataset)")

plt.xlabel("Years of Experience")

plt.ylabel("Salary(In rupees)")

plt.show()

plt.scatter(x\_test, y\_test, color="blue")

plt.plot(x\_train, x\_pred, color="red")

plt.title("Salary vs Experience(Test Dataset)")

plt.xlabel("years of Experience")

plt.ylabel("Salary(In rupees)")

plt.show()

**Output:**

**b) Logistic Regression**

**Program:**

from sklearn.metrics import accuracy\_score

from sklearn.metrics import confusion\_matrix

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

data\_set = pd.read\_csv("User\_Data.csv")

x = data\_set.iloc[:,[2, 3]].values

y = data\_set.iloc[:, 4].values

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=1/3, random\_state=0)

st\_x = StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.transform(x\_test)

model = LogisticRegression()

model.fit(x\_train, y\_train)

y\_pred = model.predict(x\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print("confusion matrix\n", cm)

print("Accuracy : ", accuracy\_score(y\_test, y\_pred))

**Output:**

confusion matrix

[[79 6]

[11 38]]

Accuracy : 0.8731343283582089

**Experiment – 5**

**Aim:** Develop a program for Bias, Variance, Remove duplicates, Cross Validation

**Program:**

import pandas as pd

data = {"A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"], "B": [

50, 40, 40, 30, 50], "C": [True, False, False, False, True]}

df = pd.DataFrame(data)

print(df)

display(df.drop\_duplicates())

**Output:**

A B C

0 TeamA 50 True

1 TeamB 40 False

2 TeamB 40 False

3 TeamC 30 False

4 TeamA 50 True

A B C

0 TeamA 50 True

1 TeamB 40 False

3 TeamC 30 False

**Cross Validation:**

**Program:**

from sklearn import datasets

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import KFold, cross\_val\_score

x, y = datasets.load\_iris(return\_X\_y=True)

clf = DecisionTreeClassifier(random\_state=0)

k\_folds = KFold(n\_splits=5)

Scores = cross\_val\_score(clf, x, y, cv=k\_folds)

print("cross validation scores:", Scores)

print("Average cv scores :", Scores.mean())

print("Number of cv scores used in Average", len(Scores))

**Output:**cross validation scores: [1. 0.96666667 0.83333333 0.93333333 0.8 ]

Average cv scores : 0.9066666666666666

Number of cv scores used in Average 5

**Bias, Variance:**

**Program:**

from mlxtend.evaluate import bias\_variance\_decomp

from sklearn.tree import DecisionTreeClassifier

from mlxtend.data import iris\_data

from sklearn.model\_selection import train\_test\_split

x, y = iris\_data()

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=0.3, random\_state=123, shuffle=True, stratify=y)

tree = DecisionTreeClassifier(random\_state=123)

avg\_expected\_loss, avg\_bias, avg\_var = bias\_variance\_decomp(

tree, x\_train, y\_train, x\_test, y\_test, loss='0-1\_loss', random\_seed=123, num\_rounds=1000)

print(f'Average Expected loss:{round(avg\_expected\_loss,4)}n')

print(f'Average Bias:{round(avg\_bias,4)}n')

print(f'Average Variance:{round(avg\_var,4)}n')

**Output:**

Average Expected loss:0.0607n

Average Bias:0.0222n

Average Variance:0.0393n

**Experiment – 6**

**Aim:** Write a program to implement Categorical Encoding, One-hot Encoding

**Program:**

from numpy import asarray

from sklearn.preprocessing import OneHotEncoder

data = asarray([['red',], ['green'], ['blue']])

print(data)

encoder = OneHotEncoder(sparse=False)

onehot = encoder.fit\_transform(data)

print(onehot)

**Output:**

[['red']

['green']

['blue']]

[[0. 0. 1.]

[0. 1. 0.]

[1. 0. 0.]]

**Experiment – 8**

**Aim:** Write a program to implement k-Nearest Neighbor algorithm to classify the iris data set. Print both correct and wrong predictions.

**Program:**

import numpy as np

import matplotlib.pyplot as mtp

import pandas as pd

data\_set = pd.read\_csv("Iris.csv")

x = data\_set.iloc[:, [2, 3]].values

y = data\_set.iloc[:, 4].values

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=0.25, random\_state=0)

from sklearn.preprocessing import StandardScaler

st\_x = StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.transform(x\_test)

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors=5, metric='minkowski', p=2)

classifier.fit(x\_train, y\_train)

y\_pred = classifier.predict(x\_test)

print(y\_pred)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

**Output:**

['Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'

'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'

'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'

'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa'

'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'

'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'

'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-versicolor'

'Iris-versicolor' 'Iris-setosa' 'Iris-virginica' 'Iris-versicolor'

'Iris-setosa' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor'

'Iris-setosa' 'Iris-versicolor']

[[13 0 0]

[ 0 16 0]

[ 0 0 9]]

**Experiment – 9**

**Aim:** Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select appropriate data set for your experiment and draw graphs.

**Program:**

import numpy as np

import matplotlib.pyplot as plt

from moepy import lowess

x = np.linspace(0, 5, num=150)

y = np.sin(x)+(np.random.normal(size=len(x)))/10

lowess\_model = lowess.Lowess()

lowess\_model.fit(x, y)

x\_pred = np.linspace(0, 5, 26)

y\_pred = lowess\_model.predict(x\_pred)

plt.plot(x\_pred, y\_pred, '--', label='Lowess', color='r', zorder=3)

plt.scatter(x, y, label='Noisy sin wave', color='m', s=10, zorder=1)

**Output:**

**Experiment – 10**

**Aim:** Assuming a set of documents that need to be classified, use the naïve Bayesian Classifier model to perform this task. Built-in Java classes/API can be used to write the program. Calculate the accuracy, precision, and recall for your data set.

**Program:**

import numpy as np

import matplotlib.pyplot as mtp

import pandas as pd

data\_set = pd.read\_csv("User\_Data.csv")

x = data\_set.iloc[:, [2, 3]].values

y = data\_set.iloc[:,4].values

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(

x, y, test\_size=0.25, random\_state=0)

from sklearn.preprocessing import StandardScaler

st\_x = StandardScaler()

x\_train = st\_x.fit\_transform(x\_train)

x\_test = st\_x.transform(x\_test)

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(x\_train, y\_train)

y\_pred = classifier.predict(x\_test)

print(y\_pred)

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

from sklearn.metrics import precision\_score

precision = precision\_score(y\_test, y\_pred)

from sklearn.metrics import recall\_score

recall = recall\_score(y\_test, y\_pred)

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test, y\_pred)

print('precision:', precision)

print('recall:', recall)

print('Accuracy:', accuracy)

**Output:**

[0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0

0 0 1 0 0 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0

0 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1]

[[65 3]

[ 7 25]]

precision: 0.8928571428571429

recall: 0.78125

Accuracy: 0.9

**Experiment – 12**

**Aim:** Exploratory Data Analysis for Classification using Pandas or Matplotlib.

**Program:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as mtp

import seaborn as sns

df = pd.read\_csv("User\_Data.csv")

print(df)

df.head()

df.tail()

df.Age.describe()

df.info()

df.Gender.value\_counts()

sns.catplot(x="Purchased", y="EstimatedSalary",

data=df, kind="box", aspect=1.5)

mtp.title("Boxplot for target vs proline")

mtp.show()

**Output:**

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

.. ... ... ... ... ...

395 15691863 Female 46 41000 1

396 15706071 Male 51 23000 1

397 15654296 Female 50 20000 1

398 15755018 Male 36 33000 0

399 15594041 Female 49 36000 1

[400 rows x 5 columns]

User ID Gender Age EstimatedSalary Purchased

0 15624510 Male 19 19000 0

1 15810944 Male 35 20000 0

2 15668575 Female 26 43000 0

3 15603246 Female 27 57000 0

4 15804002 Male 19 76000 0

User ID Gender Age EstimatedSalary Purchased

395 15691863 Female 46 41000 1

396 15706071 Male 51 23000 1

397 15654296 Female 50 20000 1

398 15755018 Male 36 33000 0

399 15594041 Female 49 36000 1

count 400.000000

mean 37.655000

std 10.482877

min 18.000000

25% 29.750000

50% 37.000000

75% 46.000000

max 60.000000

Name: Age, dtype: float64

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 400 entries, 0 to 399

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 User ID 400 non-null int64

1 Gender 400 non-null object

2 Age 400 non-null int64

3 EstimatedSalary 400 non-null int64

4 Purchased 400 non-null int64

dtypes: int64(4), object(1)

memory usage: 15.8+ KB

Female 204

Male 196

Name: Gender, dtype: int64

**Experiment – 14**

**Aim:** Write a program to Implement Support Vector Machines and Principle Component Analysis

**Program:**

import pandas as pd

from sklearn import svm

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

iris=pd.read\_csv("Iris.csv",names=['speal\_length','sepal\_width','petal\_length','petal\_width','class'])

x\_train,x\_test,y\_train,y\_test=train\_test\_split(iris.drop('class',axis=1),iris['class'],test\_size=0.3,random\_state=42)

clf=svm.svc(kernak='rbf')

clf.fit(x\_train,y\_train)

y\_pred=clf.predict(x\_test)

accuracy=accuracy\_score(y\_test,y\_pred)

print("Accuracy:",accuracy)

**Output:**

Accuracy: 0.9

**Experiment – 15**

**Aim:** Write a program to Implement Principle Component Analysis

**Program:**

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import load\_breast\_cancer

cancer = load\_breast\_cancer()

df = pd.DataFrame(cancer['data'], columns=cancer['feature\_names'])

scalar = StandardScaler()

scalar.fit(df)

scalar\_data = scalar.transform(df)

pca = PCA(n\_components=2)

pca.fit(scalar\_data)

x\_pca = pca.transform(scalar\_data)

x\_pca.shape

plt.figure(figsize=(8, 6))

plt.scatter(x\_pca[:, 0], x\_pca[:, 1], c=cancer['target'], cmap='plasma')

plt.xlabel('First Pricipal Component')

plt.ylabel('Second Principal Component')

**Output:**