99) Practical File

Q1) Extract the data from the database using python

CODE

```
import pandas as pd

data = pd.read_csv('cancer.csv')
print(data.head())
```

OUTPUT

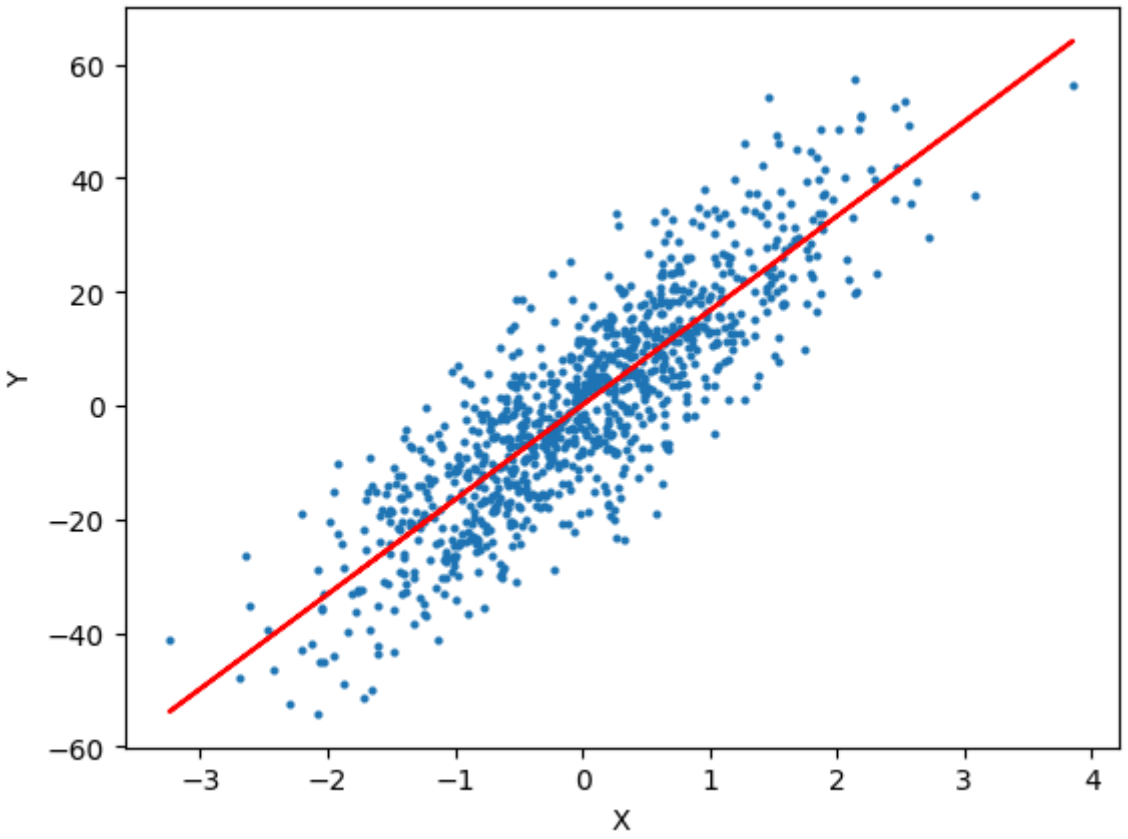
Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin
5	1	1	1	2	1	3
5	4	4	5	7	10	3
3	1	1	1	2	2	3
6	8	8	1	3	4	3
4	1	1	3	2	1	3

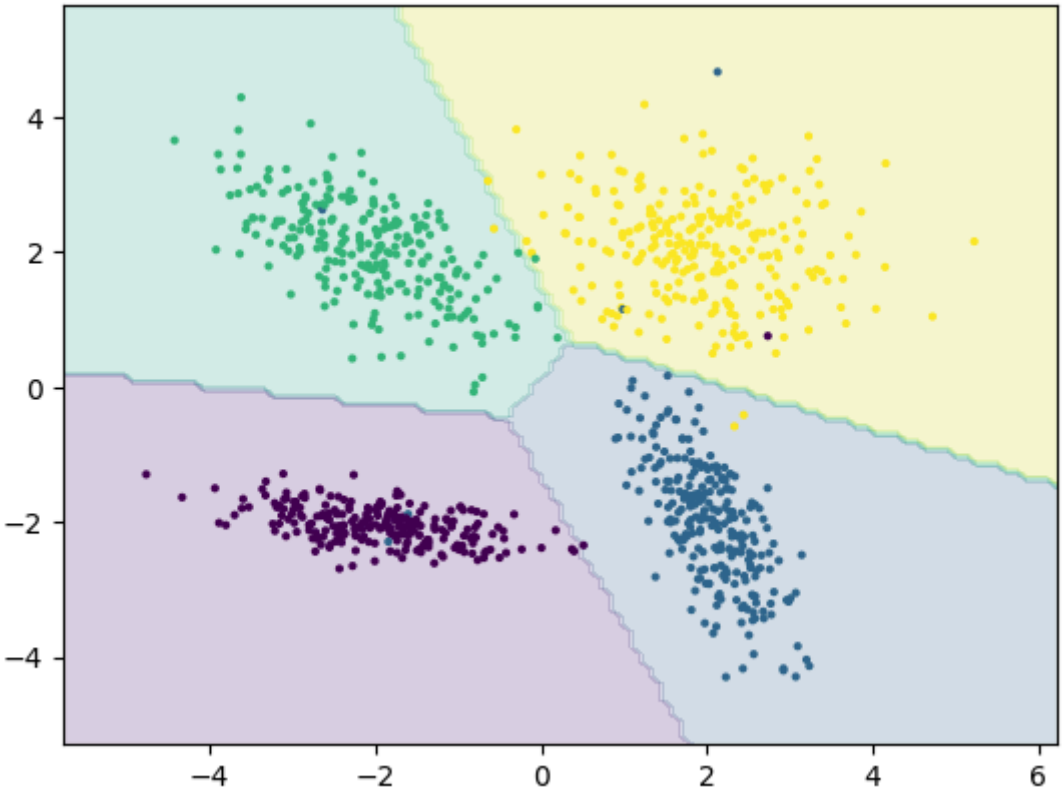
Q2) Write a program to implement linear and logistic regression

CODE

```
from sklearn import datasets
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.inspection import DecisionBoundaryDisplay

import matplotlib.pyplot as plt

plt.rcParams['lines.markersize'] = 2


# Linear Regression
X,Y = datasets.make_regression(n_samples=1000, n_features=1, noise=10)
model = LinearRegression()
model.fit(X,Y)

pred = model.predict(X)

plt.scatter(X,Y)
plt.plot(X, pred, color='red')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()


# Logistic Regression
X,Y = datasets.make_classification(n_samples=1000, n_features=2,
n_classes=4, n_clusters_per_class=1, n_redundant=0, class_sep=2)
model = LogisticRegression()
model.fit(X,Y)

fig,ax = plt.subplots(1,1)
DecisionBoundaryDisplay.from_estimator(model, X, ax=ax, alpha=0.2)
ax.scatter(X[:,0], X[:,1], c=Y)
plt.show()
```

OUTPUT





Q3) Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .csv file. Compute the accuracy of the classifier, considering few test data sets

CODE

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split, cross_val_score
import pandas as pd


df = pd.read_csv('cancer.csv')
x = df.iloc[ : , :-1 ]
y = df.iloc[ : , -1 ]
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)

model = GaussianNB()
model.fit(x_train, y_train)
accs = cross_val_score(model, x, y, cv=5)

print(f'Accuracies: {accs}')
print(f'Averaged accuracy: {round(accs.mean()*100,2)}%')
```

OUTPUT

```
Accuracies: [0.94160584 0.93430657 0.97080292 0.97058824 0.97794118]
Averaged accuracy: 95.9%
```

Q4) Write a program to implement k-nearest neighbors (KNN) and Support Vector Machine (SVM) Algorithm for classification

CODE

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix

import pandas as pd



data = pd.read_csv('cancer.csv')
x = data.iloc[:, :-1]
y = data.iloc[:, -1]



# K Nearest Neighbours
model = KNeighborsClassifier(n_neighbors=5)
model.fit(x,y)
y_pred = model.predict(x)
print('K Nearest Neighbours:')
print(confusion_matrix(y,y_pred))



# Support Vector Machine
model = SVC()
model.fit(x,y)
y_pred = model.predict(x)
print('Support Vector Machine:')
print(confusion_matrix(y,y_pred))
```

OUTPUT

```
K Nearest Neighbours:
[[435   9]
 [  4 235]]
```

```
Support Vector Machine:
[[433  11]
 [  5 234]]
```

Q5) Implement classification of a given dataset using random forest

CODE

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix

import pandas as pd



data = pd.read_csv('cancer.csv')
x = data.iloc[:, :-1]
y = data.iloc[:, -1]

model = RandomForestClassifier()
model.fit(x,y)
y_pred = model.predict(x)
print(confusion_matrix(y,y_pred))
```

OUTPUT

```
[[444   0]
 [  0 239]]
```

Q6) Build an Artificial Neural Network (ANN) by implementing the Back propagation algorithm and test the same using appropriate data sets

CODE

```
from keras.models import Sequential
from keras.layers import Dense, Input
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

import pandas as pd



df = pd.read_csv('cancer.csv')
x  = df.iloc[:, :-1]
y  = df.iloc[:, -1]
n  = x.shape[1]

# Split the data into training, testing, and validation sets
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2)
x_train, x_val,  y_train, y_val  = train_test_split(x_train, y_train,
test_size=0.2)



model = Sequential()
# Input Layer
model.add( Input(shape=(n,)) )
# Hidden Layer
model.add( Dense(4, activation='sigmoid') )
# Output Layer
model.add( Dense(1, activation='sigmoid') )

optimizer = Adam(learning_rate=0.01)
loss = 'binary_crossentropy'
```

```
model.compile(optimizer, loss=loss)
model.fit(x_train, y_train, epochs=20, batch_size=10, validation_data=
(x_val,y_val), verbose=0)



y_pred = model.predict(x_test, verbose=0)
y_pred = (y_pred > 0.5)

cm = confusion_matrix(y_test,y_pred)
print(cm)
```

OUTPUT

```
[[81  3]
 [ 3 50]]
```

Q7) Apply k-Means algorithm k-Means algorithm to cluster a set of data stored in a .csv file and evaluate its performance

CODE

```
from sklearn.cluster import KMeans
from sklearn.metrics import adjusted_rand_score

import pandas as pd



df = pd.read_csv('cancer.csv')
x  = df.iloc[:, :-1]
y  = df.iloc[:, -1]
n  = y.nunique()



model  = KMeans(n_clusters=n)
y_pred = model.fit_predict(x)
score  = adjusted_rand_score(y,y_pred)
print(score)
```

OUTPUT

```
0.8464675664733539
```

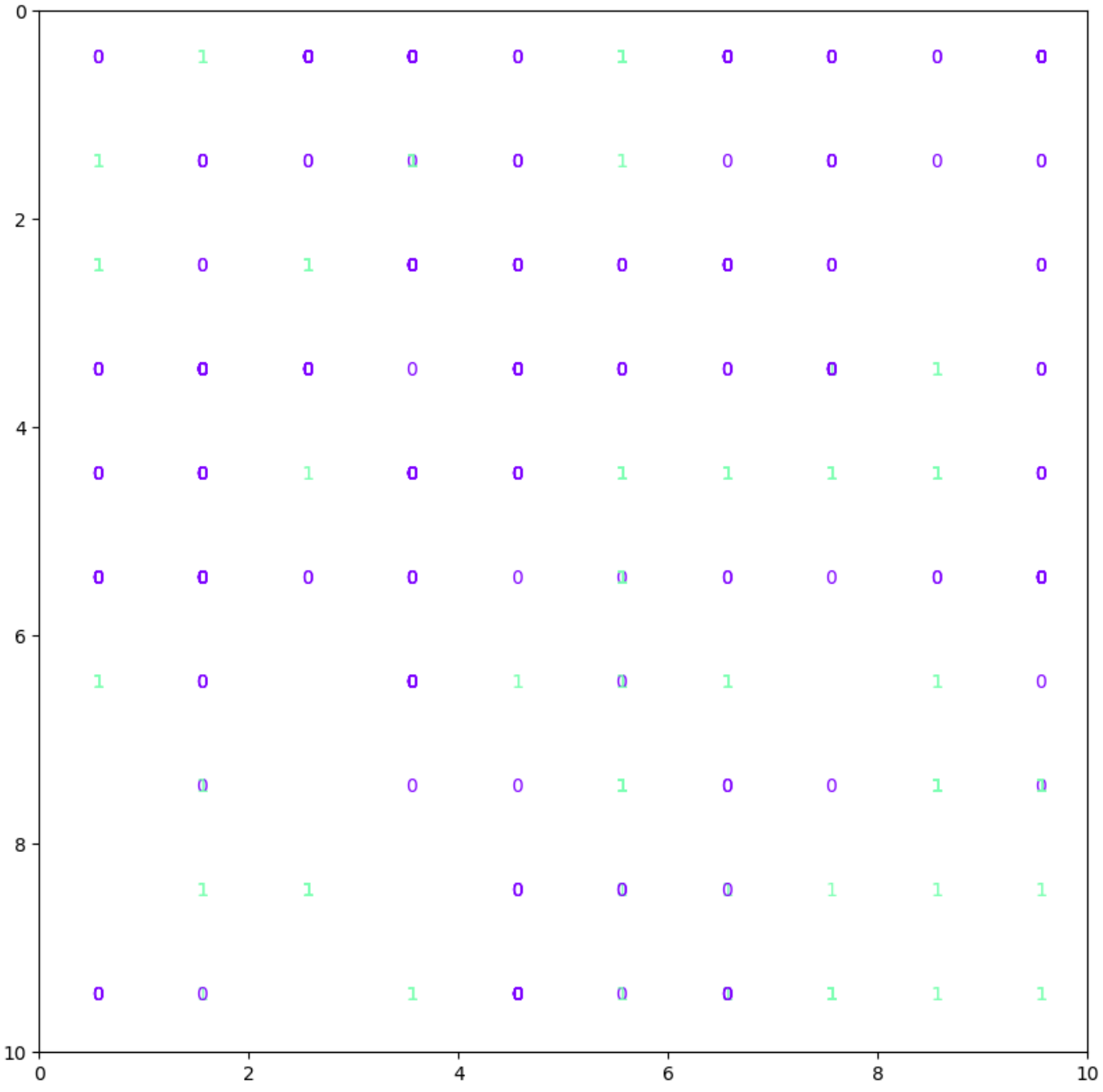
Q8) Write a program to implement Self - Organizing Map (SOM)

CODE

```
from sklearn.preprocessing import MinMaxScaler
from minisom import MiniSom

import matplotlib.pyplot as plt

import pandas as pd



data = pd.read_csv('cancer.csv')
X = data.iloc[: , :-1].values
y = data.iloc[: , -1].values

# Scale the data to [0, 1]
scaler   = MinMaxScaler()
X_scaled = scaler.fit_transform(X)



# Initialize the Self-Organizing Map
som_x = 10
som_y = 10
som = MiniSom(x=som_x, y=som_y, input_len=X.shape[1])
# Randomly initialize the weights
som.random_weights_init(X_scaled)
# Train the SOM
som.train_random(X_scaled, num_iteration=100)



# Visualize the results
plt.figure(figsize=(10, 10))
for i, x in enumerate(X_scaled):
    win = som.winner(x)  # Get the winning node
    x_val = win[0] + 0.5
    y_val = win[1] + 0.5
    plt.text(x_val, y_val, str(y[i]), color=plt.cm.rainbow(y[i]/2))
```

```
plt.axis([0, som_x, 0, som_y])
plt.gca().invert_yaxis()
plt.show()
```

OUTPUT



Q9) Write a program for empirical comparison of different supervised learning algorithms

CODE

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB


import pandas as pd


df = pd.read_csv('cancer.csv')
x  = df.iloc[:, :-1]
y  = df.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=0)


models = {
    'Logistic Regression': LogisticRegression(),
    'SVM': SVC(),
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'Naive Bayes': GaussianNB()
}


for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
```

```
    score  = f1_score(y_test,y_pred)
    print(f'{name}: {score}')
```

OUTPUT

```
Logistic Regression: 0.94
SVM: 0.9514563106796117
Decision Tree: 0.9019607843137255
Random Forest: 0.9607843137254902
Naive Bayes: 0.9345794392523364
```

Q10) Write a program for empirical comparison of different unsupervised learning algorithms

CODE

```
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import adjusted_rand_score

import pandas as pd



df = pd.read_csv('cancer.csv')
x  = df.iloc[:, :-1]
y  = df.iloc[:, -1]
n  = y.nunique()


models = {
    'KMeans': KMeans(n),
    'AgglomerativeClustering': AgglomerativeClustering(n),
    'DBSCAN': DBSCAN()
}


for name,model in models.items():
    y_pred = model.fit_predict(x)
    score  = adjusted_rand_score(y,y_pred)
    print(f'{name}: {score}')
```

OUTPUT

```
KMeans: 0.8464675664733539
AgglomerativeClustering: 0.8689991723757481
DBSCAN: -0.07424593466273072
```