1. Implement the Naive Bayes Classifier on the below given dataset. Test record for this given dataset is (Rainy, Cool, Normal, True). Also test the same on a large dataset with a sample test record.

| Outlook | Temperature | Humidity | Windy | PlayGolf |
|---|---|---|---|---|
| Rainy | Hot | High | False | No |
| Rainy | Hot | High | True | No |
| Overcast | Hot | High | False | Yes |
| Sunny | Mild | High | False | Yes |
| Sunny | Cool | Normal | False | Yes |
| Sunny | Cool | Normal | True | No |
| Overcast | Cool | Normal | True | Yes |
| Rainy | Mild | High | False | No |
| Rainy | Cool | Normal | False | Yes |
| Sunny | Mild | Normal | False | Yes |
| Rainy | Mild | Normal | True | Yes |
| Overcast | Mild | High | True | Yes |
| Overcast | Hot | Normal | False | Yes |
| Sunny | Mild | High | True | No |

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

data = [[1, 1, 1, 0],
        [1, 1, 1, 1],
        [0, 1, 1, 0],
        [2, 0, 1, 0],
        [2, 2, 0, 0],
        [2, 2, 0, 1],
        [0, 2, 0, 1],
        [1, 0, 1, 0],
        [1, 2, 0, 0],
        [2, 0, 0, 0],
        [1, 0, 0, 1],
        [0, 0, 1, 1],
        [0, 1, 0, 0],
        [2, 0, 1, 1]]

target = [0,0,1,1,1,1,0,1,1,1,1,1,1,0]

# splitting data and target into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.4)

# training the model on training set
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("The accuracy is:", accuracy)
```

```
The accuracy is: 0.9166666666666666
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# load the iris dataset
data = load_iris()

# splitting data and target into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.4)

# training the model on training set
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("The accuracy is:", accuracy)
```

```
The accuracy is: 0.95
```

2.1 Implement a Neural Network Classifier (Back Propagation Network(BPN)) involving at least one hidden layer and trace the BPN algorithm assuming a target class for one epoch. Compare the evaluation of the network on a test set by varying the number of hidden nodes/layers.

- 1 hidden layer

```
# Import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the model architecture
model = Sequential()
model.add(Dense(10, input_dim=30, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=1)

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: {:.2f}".format(accuracy * 100))
```

```
Epoch 1/50
Epoch 2/50
Epoch 3/50
Epoch 4/50
Epoch 5/50
Epoch 6/50
Epoch 7/50
Epoch 8/50
Epoch 9/50
Epoch 10/50
Epoch 11/50
Epoch 12/50
Epoch 13/50
Epoch 14/50
Epoch 15/50
Epoch 16/50
Epoch 17/50
Epoch 18/50
Epoch 19/50
Epoch 20/50
Epoch 21/50
Epoch 22/50
Epoch 23/50
Epoch 24/50
Epoch 25/50
Epoch 26/50
Epoch 27/50
Epoch 28/50
Epoch 29/50
Epoch 30/50
Epoch 31/50
Epoch 32/50
Epoch 33/50
Epoch 34/50
Epoch 35/50
Epoch 36/50
Epoch 37/50
Epoch 38/50
Epoch 39/50
Epoch 40/50
Epoch 41/50
Epoch 42/50
Epoch 43/50
Epoch 44/50
Epoch 45/50
Epoch 46/50
Epoch 47/50
Epoch 48/50
Epoch 49/50
Epoch 50/50
Accuracy: 97.37
```

- 3 hidden layer

```
# Import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the model architecture with three hidden layers
model = Sequential()
model.add(Dense(10, input_dim=30, activation='relu'))
model.add(Dense(5), activation='relu'))
model.add(Dense(3), activation='relu'))
model.add(Dense(1), activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=1)

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: {:.2f}".format(accuracy * 100))
```

```
Epoch 1/50
Epoch 2/50
Epoch 3/50
Epoch 4/50
Epoch 5/50
Epoch 6/50
Epoch 7/50
Epoch 8/50
Epoch 9/50
Epoch 10/50
Epoch 11/50
Epoch 12/50
Epoch 13/50
Epoch 14/50
Epoch 15/50
Epoch 16/50
Epoch 17/50
Epoch 18/50
Epoch 19/50
Epoch 20/50
Epoch 21/50
Epoch 22/50
Epoch 23/50
Epoch 24/50
Epoch 25/50
Epoch 26/50
Epoch 27/50
Epoch 28/50
Epoch 29/50
Epoch 30/50
Epoch 31/50
Epoch 32/50
Epoch 33/50
Epoch 34/50
Epoch 35/50
Epoch 36/50
Epoch 37/50
Epoch 38/50
Epoch 39/50
Epoch 40/50
Epoch 41/50
Epoch 42/50
Epoch 43/50
Epoch 44/50
Epoch 45/50
Epoch 46/50
Epoch 47/50
Epoch 48/50
Epoch 49/50
Epoch 50/50
Accuracy: 97.37
```

- 5 hidden layer

```
# Import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the model architecture with five hidden layers
model = Sequential()
model.add(Dense(10, input_dim=30, activation='relu'))
model.add(Dense(5), activation='relu'))
model.add(Dense(3), activation='relu'))
model.add(Dense(2), activation='relu'))
model.add(Dense(1), activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=10, verbose=1)

# Evaluate the model on the test set
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: {:.2f}".format(accuracy * 100)
```

```
Epoch 1/50
Epoch 2/50
Epoch 3/50
Epoch 4/50
Epoch 5/50
Epoch 6/50
Epoch 7/50
Epoch 8/50
Epoch 9/50
Epoch 10/50
Epoch 11/50
Epoch 12/50
Epoch 13/50
Epoch 14/50
Epoch 15/50
Epoch 16/50
Epoch 17/50
Epoch 18/50
Epoch 19/50
Epoch 20/50
Epoch 21/50
Epoch 22/50
Epoch 23/50
Epoch 24/50
Epoch 25/50
Epoch 26/50
Epoch 27/50
Epoch 28/50
Epoch 29/50
Epoch 30/50
Epoch 31/50
Epoch 32/50
Epoch 33/50
Epoch 34/50
Epoch 35/50
Epoch 36/50
Epoch 37/50
Epoch 38/50
Epoch 39/50
Epoch 40/50
Epoch 41/50
Epoch 42/50
Epoch 43/50
Epoch 44/50
Epoch 45/50
Epoch 46/50
Epoch 47/50
Epoch 48/50
Epoch 49/50
Epoch 50/50
Accuracy: 97.37
```

- 2.2 Also simulate the behaviour of logic gates like OR gate, AND gate using Neural networks.

- OR Gate

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

# Define the input data and labels for the OR gate
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 1, 1, 1])

# Define the neural network architecture
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))
model.add(Dense(1), activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=100, verbose=0)

# Test the model on the input data
print(model.predict(np.array([[0, 0], [0, 1], [1, 0], [1, 1]])))
```

```
1/1 [==============================] - 0s 106ms/step
[[0.06]
 [0.94]
 [0.94]
 [0.94]]
```

- AND Gate

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

# Define the input data and labels for the AND gate
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 0], [0], [1]])

# Define the neural network architecture
model = Sequential()
model.add(Dense(1, input_dim=2, activation='sigmoid'))
model.add(Dense(1), activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=100, verbose=0)

# Test the model on the input data
print(model.predict(np.array([[0, 0], [0, 1], [1, 0], [1, 1]])))
```

```
1/1 [==============================] - 0s 106ms/step
[[0.06]
 [0.06]
 [0.06]
 [0.94]]
```

Colab paid products - Cancel contracts here