Program 1: Simple Linear Regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Load dataset
data = pd.read_csv('headbrain.csv')
X = data['Head Size(cm^3)'].values.reshape(-1, 1)
Y = data['Brain Weight(grams)'].values

# Train model
model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)

# Plot results
plt.scatter(X, Y, color='red', label='Data points')
plt.plot(X, Y_pred, color='blue', label='Regression line')
plt.xlabel('Head Size (cm^3)')
plt.ylabel('Brain Weight (grams)')
plt.legend()
plt.show()

# R^2 score
print(f'R^2 Score: {r2_score(Y, Y_pred)}')
```

Program 2: Multiple Linear Regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Load dataset
data = pd.read_csv('housing_prices.csv')
X = data[['AREA', 'FLOOR', 'ROOM']].values
y = data['PRICE'].values

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate
print(f'Train R^2: {model.score(X_train, y_train)}')
print(f'Test R^2: {model.score(X_test, y_test)}')
```

Program 3: Naive Bayes Classifier

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('breast_cancer.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = GaussianNB()
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

Program 4: Decision Tree Classifier

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

# Load dataset
data = pd.read_csv('breast_cancer.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

Program 5: K-Means and Hierarchical Clustering

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage

# Load dataset for K-Means
kmeans_data = pd.read_csv('ch1ex1.csv').values
kmeans_model = KMeans(n_clusters=3)
kmeans_model.fit(kmeans_data)
labels = kmeans_model.labels_

# Plot K-Means
plt.scatter(kmeans_data[:, 0], kmeans_data[:, 1], c=labels)
plt.title("K-Means Clustering")
plt.show()

# Load dataset for Hierarchical Clustering
hierarchical_data = pd.read_csv('seeds-less-rows.csv').drop('grain_variety', axis=1).values
linkage_matrix = linkage(hierarchical_data, method='complete')

# Plot Dendrogram
dendrogram(linkage_matrix)
plt.title("Hierarchical Clustering Dendrogram")
plt.show()
```

Program 6: Neural Network on Pima Indians Diabetes Dataset

```
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split

# Load dataset
data = pd.read_csv('pima-indians-diabetes.csv')
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define model
model = Sequential([
    Dense(8, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile and train
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test))
```

Program 7: CNN on MNIST Dataset

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.datasets import mnist
from keras.utils import to_categorical

# Load dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(-1, 28, 28, 1) / 255.0
X_test = X_test.reshape(-1, 28, 28, 1) / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Define model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile and train
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test))
```

Program 8: Simple RNN on IMDB Dataset

```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN, Dense
from keras.datasets import imdb
from keras.preprocessing.sequence import pad_sequences

# Load dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
X_train = pad_sequences(X_train, maxlen=500)
X_test = pad_sequences(X_test, maxlen=500)

# Define model
model = Sequential([
    Embedding(10000, 32),
    SimpleRNN(32),
    Dense(1, activation='sigmoid')
])

# Compile and train
model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test))
```