For the wine data set in the below link: https://archive.ics.uci.edu/dataset/109/wine what different models can i fit? only list me the models

ChatGPT said:

Here’s a list of different models you can fit on the Wine dataset:

1. Logistic Regression
2. k-Nearest Neighbors (k-NN)
3. Support Vector Machine (SVM)
4. Decision Tree
5. Random Forest
6. Gradient Boosting (e.g., XGBoost, LightGBM)
7. Naive Bayes
8. Linear Discriminant Analysis (LDA)
9. Quadratic Discriminant Analysis (QDA)
10. Multi-layer Perceptron (Neural Network)
11. Ridge Classifier
12. Bagging Classifier
13. Extra Trees Classifier

Let me know if you'd like a comparison of model performance, code examples, or a visual workflow.

You said:

can i build a simple neural network?

ChatGPT said:

Yes, you can absolutely build a simple neural network for the Wine dataset. Since it's a multiclass classification problem with 3 classes and 13 numerical features, a small feedforward neural network (multi-layer perceptron) with:

- **Input layer:** 13 neurons (one for each feature)
- **Hidden layer(s):** 1 or 2 layers with 8–32 neurons (ReLU activation)
- **Output layer:** 3 neurons with softmax activation

would be sufficient.

Let me know if you'd like a PyTorch example, TensorFlow/Keras version, or a training visualization.

You said:

build me a simple NN. Use 50 Epochs at max.

ChatGPT said:

Here’s a simple neural network using **Keras (TensorFlow)** to classify the Wine dataset, training for **up to 50 epochs**:

```
import numpy as np
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load data
data = load_wine()
X = data.data
y = data.target

# Preprocess
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
y_cat = to_categorical(y)  # one-hot encoding for multiclass

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_cat, test_size=0.2, random_state=42)

# Build model
model = Sequential([
    Dense(16, input_shape=(X.shape[1],), activation='relu'),
    Dense(8, activation='relu'),
    Dense(3, activation='softmax')  # 3 classes
])


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


# Train
history = model.fit(X_train, y_train, epochs=50, validation_data=(X_test, y_test), verbose=1)


# Evaluate
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print(f'Test Accuracy: {test_acc:.4f}')
```

Let me know if you want a plot of the training history, a PyTorch version, or model export options.

You said:

use PyTorch

ChatGPT said:

Here’s a simple neural network built with **PyTorch** for the Wine dataset, trained for **50 epochs max**:

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from torch.utils.data import TensorDataset, DataLoader


# Load and preprocess data
data = load_wine()
X = data.data
y = data.target
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

X_train = torch.tensor(X_train, dtype=torch.float32)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
y_test = torch.tensor(y_test, dtype=torch.long)

train_ds = TensorDataset(X_train, y_train)
test_ds = TensorDataset(X_test, y_test)

train_loader = DataLoader(train_ds, batch_size=16, shuffle=True)
test_loader = DataLoader(test_ds, batch_size=16)

# Define model
class WineNN(nn.Module):
    def __init__(self):
        super(WineNN, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(13, 16),
            nn.ReLU(),
            nn.Linear(16, 8),
            nn.ReLU(),
            nn.Linear(8, 3)
        )

    def forward(self, x):
        return self.model(x)
```

```
model = WineNN()

# Training setup
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
epochs = 50

# Training loop
for epoch in range(epochs):
    model.train()
    for xb, yb in train_loader:
        preds = model(xb)
        loss = criterion(preds, yb)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

# Evaluation
model.eval()
correct = 0
total = 0
with torch.no_grad():
    for xb, yb in test_loader:
        preds = model(xb)
        _, predicted = torch.max(preds, 1)
        total += yb.size(0)
        correct += (predicted == yb).sum().item()

print(f'Test Accuracy: {correct / total:.4f}')
```

Let me know if you'd like to add validation accuracy tracking, plot training loss, or apply early stopping.

You said:

Print me loss and accuracy for every 10 epochs.

ChatGPT said:

Here's the modified PyTorch code that **prints loss and accuracy every 10 epochs** during training:

```
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from torch.utils.data import TensorDataset, DataLoader

# Load and preprocess data
data = load_wine()
X = data.data
y = data.target

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

X_train = torch.tensor(X_train, dtype=torch.float32)
X_test = torch.tensor(X_test, dtype=torch.float32)
y_train = torch.tensor(y_train, dtype=torch.long)
y_test = torch.tensor(y_test, dtype=torch.long)
```

```
train_ds = TensorDataset(X_train, y_train)
test_ds = TensorDataset(X_test, y_test)

train_loader = DataLoader(train_ds, batch_size=16, shuffle=True)
test_loader = DataLoader(test_ds, batch_size=16)

# Define model
class WineNN(nn.Module):
    def __init__(self):
        super(WineNN, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(13, 16),
            nn.ReLU(),
            nn.Linear(16, 8),
            nn.ReLU(),
            nn.Linear(8, 3)
        )

    def forward(self, x):
        return self.model(x)

model = WineNN()

# Training setup
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
epochs = 50

# Training loop
for epoch in range(1, epochs + 1):
    model.train()
    running_loss = 0.0
    for xb, yb in train_loader:
        preds = model(xb)
        loss = criterion(preds, yb)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    # Print every 10 epochs
    if epoch % 10 == 0:
        model.eval()
        correct = 0
        total = 0
        with torch.no_grad():
            for xb, yb in test_loader:
                preds = model(xb)
                _, predicted = torch.max(preds, 1)
                total += yb.size(0)
                correct += (predicted == yb).sum().item()
        avg_loss = running_loss / len(train_loader)
        accuracy = correct / total
        print(f"Epoch {epoch:2d}: Loss = {avg_loss:.4f}, Accuracy = {accuracy:.4f}")
```

Let me know if you’d like to see a training curve plot, add learning rate decay, or compare with a baseline model.