**Random Forest**

python

Copy code

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, mean_absolute_error

# Assuming X is the feature matrix and y is the target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f'Random Forest Accuracy: {accuracy}')
print(f'Mean Absolute Error: {mae}')
```

**K-Nearest Neighbors (KNN)**

python

Copy code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import mean_squared_error
import numpy as np

# Initialize and train the KNN model
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)

# Make predictions
y_pred = knn_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f'KNN Accuracy: {accuracy}')
print(f'Root Mean Squared Error: {rmse}')
```

**Neural Network**

python

Copy code

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler

# Scaling the data
scaler = MinMaxScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and build the Neural Network model
nn_model = Sequential()
nn_model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
nn_model.add(Dense(32, activation='relu'))
nn_model.add(Dense(1, activation='linear'))  # For regression

# Compile the model
nn_model.compile(loss='mean_absolute_error', optimizer='adam')

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

# Make predictions
y_pred = nn_model.predict(X_test_scaled)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)

print(f'Neural Network MAE: {mae}')
```