Project: Handwritten digit recognition using MNIST dataset

Problem statement:

Given an image of a handwritten digit, the task is to build a model that can accurately predict the digit.

Dataset:

The MNIST dataset is a widely used dataset for handwritten digit recognition and contains 60,000 training images and 10,000 testing images. The images are 28x28 grayscale images of handwritten digits ranging from 0 to 9.

Model:

One simple approach to solve this problem is to use a multi-layer Perceptron (MLP) model with a softmax activation function in the output layer.

Steps:

1. Load the MNIST dataset using a popular Python library such as TensorFlow or Keras.
2. Preprocess the data by normalizing the pixel values and converting the label values into one-hot encodings.
3. Split the data into training and testing sets.
4. Train the MLP model on the training data.
5. Evaluate the model on the testing data and calculate its accuracy.
6. Use the trained model to make predictions on new images of handwritten digits.

Implementation:

Here's a sample implementation of this project using TensorFlow and Keras:

```
import tensorflow as tf

from tensorflow import keras



# Load MNIST dataset
```

```
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()


# Preprocess data
x_train = x_train / 255.0
x_test = x_test / 255.0
y_train = keras.utils.to_categorical(y_train, 10)
y_test = keras.utils.to_categorical(y_test, 10)


# Reshape data
x_train = x_train.reshape(-1, 28 * 28)
x_test = x_test.reshape(-1, 28 * 28)


# Build model
model = keras.Sequential([
    keras.layers.Dense(512, activation='relu', input_shape=(28 * 28,)),
    keras.layers.Dense(10, activation='softmax')
])


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


# Train model
```

```
model.fit(x_train, y_train, batch_size=128, epochs=10, validation_data=(x_test, y_test))



# Evaluate model

test_loss, test_acc = model.evaluate(x_test, y_test)

print('Test accuracy:', test_acc)
```

This implementation should give you an accuracy of around 98% on the MNIST testing data.