```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
from sklearn.metrics import accuracy_score, confusion_matrix
import joblib

# Load the datasets
train_data = pd.read_csv(r'mnist_train.csv')
test_data = pd.read_csv(r'mnist_test.csv')

# Explore the dataset
print("Shape of Training Data:", train_data.shape)
print("\nShape of Test Data:", test_data.shape)

# Identify unique classes
unique_classes = train_data['label'].unique()
num_classes = len(unique_classes)
print("\nNumber of Unique Classes:", num_classes)
```

```python
# Extract features and labels
X_train = train_data.iloc[:, 1:] / 255.0  # Normalize pixel values
y_train = train_data.iloc[:, 0]

X_test = test_data.iloc[:, 1:] / 255.0
y_test = test_data.iloc[:, 0]

num_features = X_train.shape[1]
print("\nNumber of features: ", num_features)

print("-----------------------------------------------------------------------")

# Check for missing values in the training dataset
missing_values_train = X_train.isnull().sum()
print("\nMissing values in X_train:\n", missing_values_train)

print("-----------------------------------------------------------------------")

# Check for missing values in the test dataset
missing_values_test = X_test.isnull().sum()
print("\nMissing values in X_test:\n", missing_values_test)

print("-----------------------------------------------------------------------")

# Reshape images
image_shape = (28, 28)
X_train = X_train.values.reshape(-1, *image_shape)  # Reshape to 28x28
X_test = X_test.values.reshape(-1, *image_shape)
```

```python
# Visualize some images from the training dataset
plt.figure(figsize=(10, 10))
for i in range(9):
    plt.subplot(3, 3, i + 1)
    plt.imshow(X_train[i], cmap='gray')
    plt.title(f"Label: {y_train.iloc[i]}")
    plt.axis('off')
print('\nVisualize some images from the dataset \n')
plt.show()

# Split dataset into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

print("----------------------------------------------------------------------")

# Print shapes of processed datasets
print("\nShapes after Preprocessing:")
print("X_train:", X_train.shape)
print("y_train:", y_train.shape)
print("X_val:", X_val.shape)
print("y_val:", y_val.shape)
print("X_test:", X_test.shape)
print("y_test:", y_test.shape)
print("----------------------------------------------------------------------")
```

```python
# Handle missing values using an imputer transformer
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train.reshape(X_train.shape[0], -1))
X_val = imputer.transform(X_val.reshape(X_val.shape[0], -1))
X_test = imputer.transform(X_test.reshape(X_test.shape[0], -1))

# Initial Experiment: K-NN classification
knn = KNeighborsClassifier()
param_grid = {'n_neighbors': [3, 5, 7]}
grid_knn = GridSearchCV(knn, param_grid, cv=5)
grid_knn.fit(X_train, y_train)
best_knn = grid_knn.best_estimator_
knn_predictions = best_knn.predict(X_val)
knn_accuracy = accuracy_score(y_val, knn_predictions)
print("\nK-NN Accuracy on Validation Data:", knn_accuracy)

print("-----------------------------------------------------------------------")

# Subsequent Experiment: ANN
num_hidden_neurons = [64, 128]
learning_rates = [0.001, 0.01]
batch_sizes = [32, 64]
best_ann_accuracy = 0.0
best_ann_model = None
```

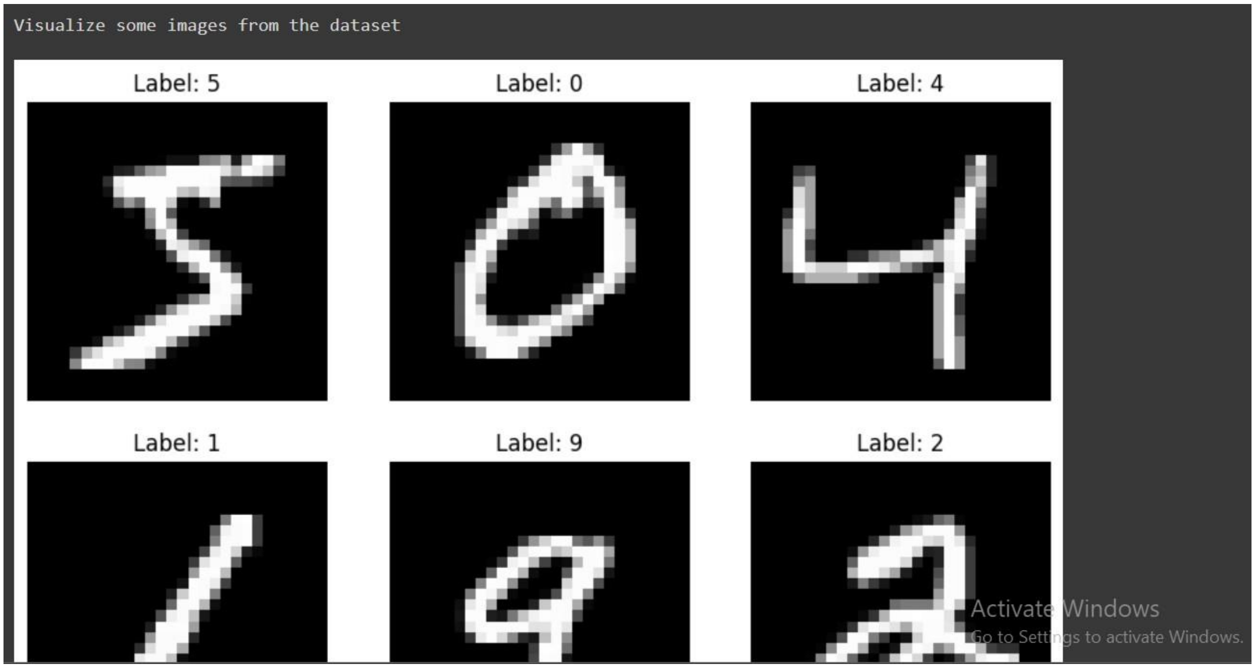
```python
for hidden_neurons in num_hidden_neurons:
    for learning_rate in learning_rates:
        for batch_size in batch_sizes:
            ann_model = Sequential()
            ann_model.add(Dense(hidden_neurons, activation='relu', input_shape=(784,)))
            ann_model.add(Dense(num_classes, activation='softmax'))

            ann_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
                              loss='sparse_categorical_crossentropy', metrics=['accuracy'])

            ann_model.fit(X_train, y_train, batch_size=batch_size, epochs=10, verbose=0)
            ann_predictions = np.argmax(ann_model.predict(X_val), axis=-1)
            ann_accuracy = accuracy_score(y_val, ann_predictions)

            print(f"ANN Accuracy (hidden neurons={hidden_neurons}, learning rate={learning_rate}, "
                  f"batch size={batch_size}): {ann_accuracy}")

            if ann_accuracy > best_ann_accuracy:
                best_ann_accuracy = ann_accuracy
                best_ann_model = ann_model

print("----------------------------------------------------------------------")

# Compare outcomes of experiments
print('\nComparison of Outcomes:')
print("K-NN Accuracy on Validation Data:", knn_accuracy)
print("Best ANN Accuracy on Validation Data:", best_ann_accuracy)
```

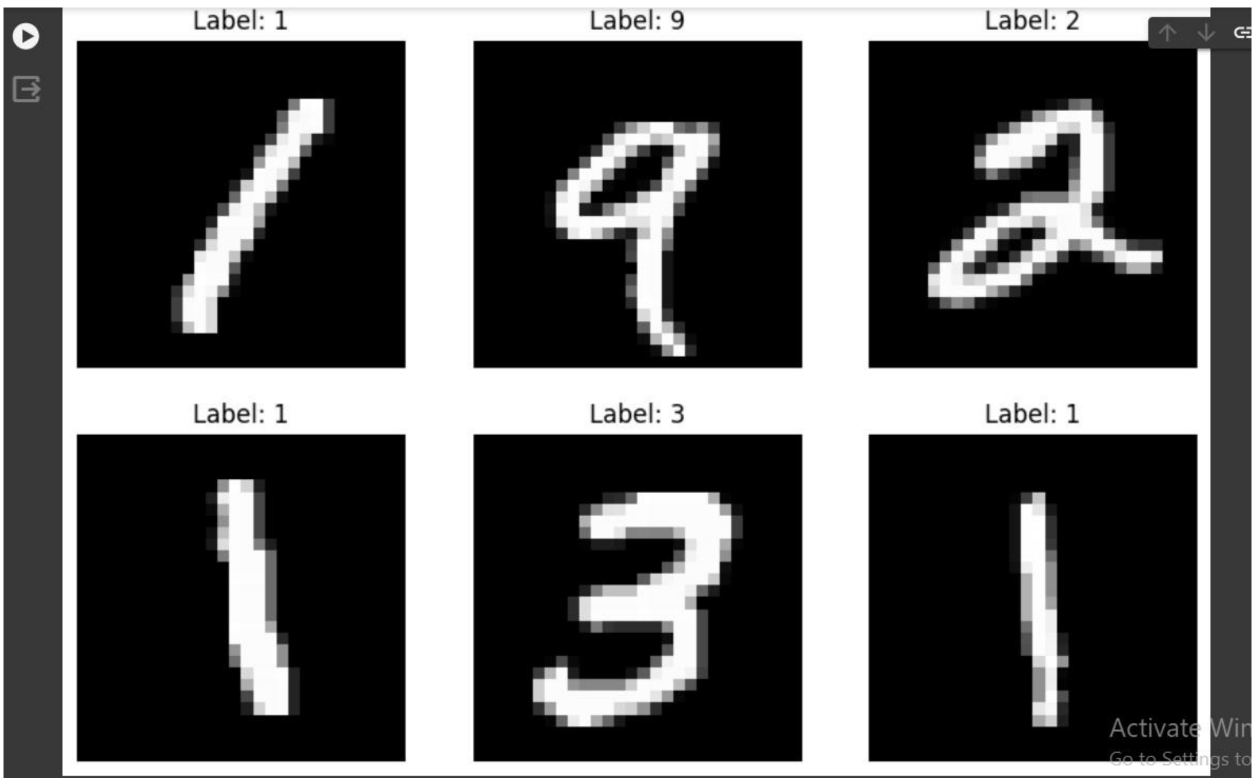


```python
# Get confusion matrix of the best model
best_ann_predictions = np.argmax(best_ann_model.predict(X_val), axis=-1)
confusion_mat = confusion_matrix(y_val, best_ann_predictions)
print("\nConfusion Matrix:")
print(confusion_mat)
print("----------------------------------------------------------------------")

# Save the best model
joblib.dump(best_ann_model, 'best_ann_model.pkl')

# Reload the best model from a separate file
loaded_model = joblib.load('best_ann_model.pkl')

# Apply the best model on the testing data
test_predictions = np.argmax(loaded_model.predict(X_test), axis=-1)
test_accuracy = accuracy_score(y_test, test_predictions)
print("\nTesting Accuracy using the Best Model:", test_accuracy)
```

```
Shape of Training Data: (55086, 785)

Shape of Test Data: (10000, 785)

Number of Unique Classes: 10

Number of features:  784
----------------------------------------------------------------------

Missing values in X_train:
 1x1      0
1x2      0
1x3      0
1x4      0
1x5      0
        ..
28x24    1
28x25    1
28x26    1
28x27    1
28x28    1
Length: 784, dtype: int64
----------------------------------------------------------------------
```

```
Missing values in X_test:
 1x1      0
1x2      0
1x3      0
1x4      0
1x5      0
        ..
28x24    0
28x25    0
28x26    0
28x27    0
28x28    0
Length: 784, dtype: int64
----------------------------------------------------------------------
```


Visualize some images from the dataset
Label: 5
Label: 0
Label: 4
Label: 1
Label: 9
Label: 2
Activate Windows
Go to Settings to activate Windows.


Label: 1
Label: 9
Label: 2
Label: 1
Label: 3
Label: 1

```
Shapes after Preprocessing:
X_train: (44068, 28, 28)
y_train: (44068,)
X_val: (11018, 28, 28)
y_val: (11018,)
X_test: (10000, 28, 28)
y_test: (10000,)
----------------------------------------------------------------------

K-NN Accuracy on Validation Data: 0.9694136866944999
----------------------------------------------------------------------
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=64, learning rate=0.001, batch size=32): 0.9673261935015429
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=64, learning rate=0.001, batch size=64): 0.963605009983663
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=64, learning rate=0.01, batch size=32): 0.9513523325467417
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=64, learning rate=0.01, batch size=64): 0.9609729533490652
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=128, learning rate=0.001, batch size=32): 0.9721365039027047
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=128, learning rate=0.001, batch size=64): 0.9732256307859866
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=128, learning rate=0.01, batch size=32): 0.9576148121256126
345/345 [==============================] - 1s 2ms/step
ANN Accuracy (hidden neurons=128, learning rate=0.01, batch size=64): 0.9601561081866037
----------------------------------------------------------------------
```

```
Comparison of Outcomes:
K-NN Accuracy on Validation Data: 0.9694136866944999
Best ANN Accuracy on Validation Data: 0.9732256307859866
----------------------------------------------------------------------
345/345 [==============================] - 1s 2ms/step

Confusion Matrix:
[[1121    0    2    1    3    1    0    1    0    3]
 [   1 1173    2    1    2    0    2    3    0    0]
 [   2    1 1059    3    0    1    4   11    1    2]
 [   1    1   15 1098    1    9    0   10    7    5]
 [   1    0    1    1 1031    0    2    3    0   13]
 [   4    2    0   12    1  981    4    3    1    0]
 [   4    1    1    0    2   10 1086    0    2    0]
 [   0    5   10    0    4    0    0 1151    0    5]
 [   4    8    7   13    4   10    5    5  965    9]
 [   5    2    0    3   16    4    0   11    1 1058]]
----------------------------------------------------------------------
313/313 [==============================] - 1s 2ms/step

Testing Accuracy using the Best Model: 0.976
```