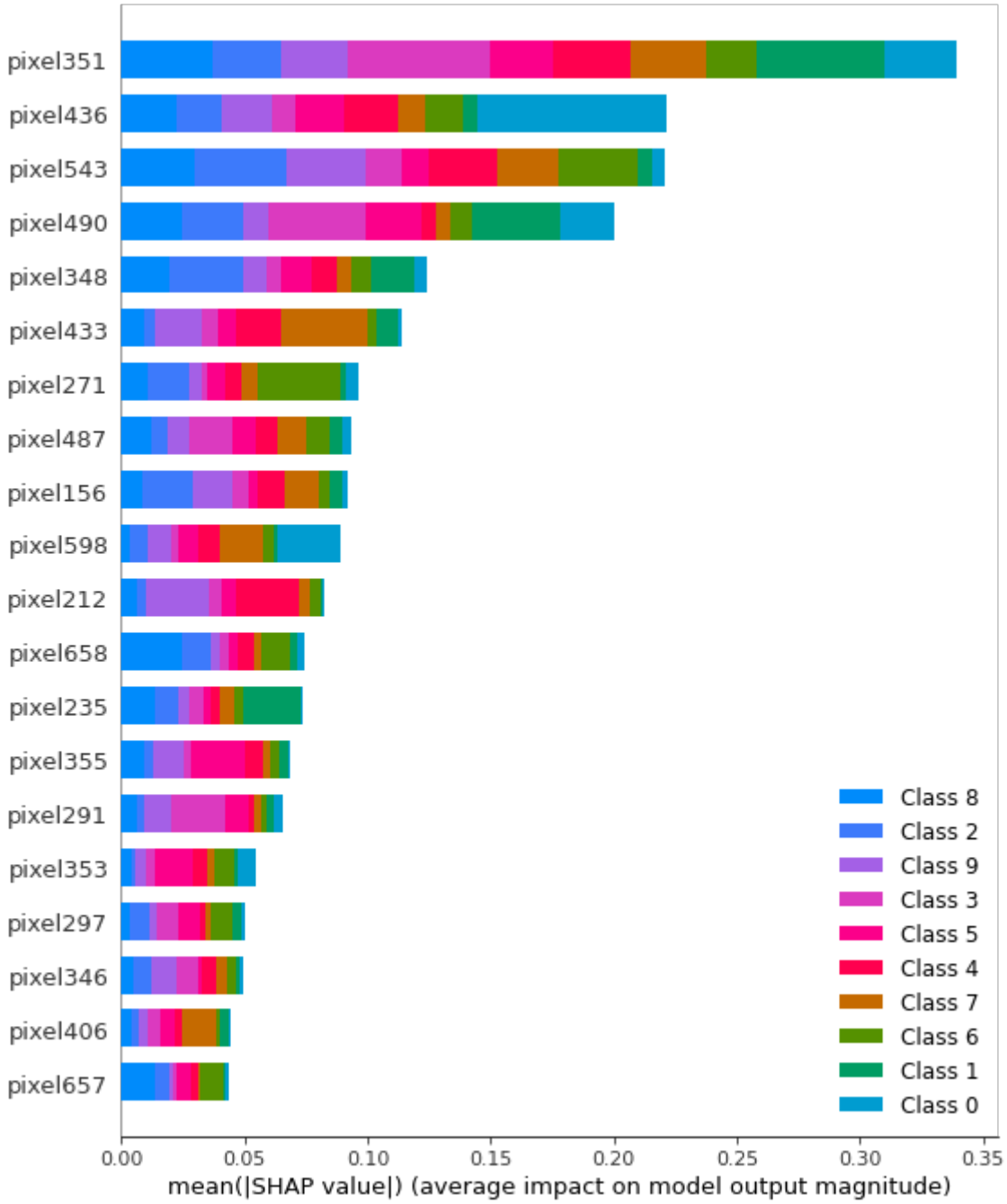
Training the Sklearn model and explaining with SHAP. Plot shows the most important features.

In [1]:

```
import shap
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Load MNIST dataset
mnist = fetch_openml('mnist_784', version=1)
X = mnist.data.astype('int')
y = mnist.target.astype('int')

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Create a decision tree classifier
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

# Use SHAP to explain the model's predictions
explainer = shap.Explainer(clf)
shap_values = explainer.shap_values(X_test)

# Summary plot of SHAP values
shap.summary_plot(shap_values, X_test, feature_names=mnist.feature_names
```

```
/usr/local/lib/python3.7/dist-packages/numba/core/errors.py:149: UserWarn
ing: Insufficiently recent colorama version found. Numba requires coloram
a >= 0.3.9
  warnings.warn(msg)
```



In [2]:
```
#Saving the model for future use
import pickle
saved_clf = pickle.dumps(clf)
```

In [3]:
```
# Initialize the SHAP
shap.initjs()
```

visualize the first 500 prediction's explanation with a force plot

In [4]:
```
shap.force_plot(explainer.expected_value[0], shap_values[0][:500,:])
```

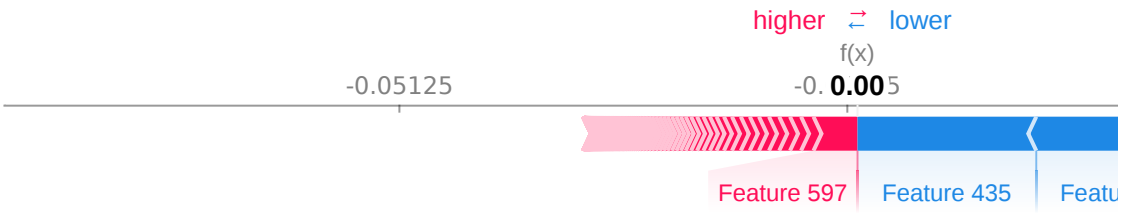
Out[4]:

visualize the first prediction's explanation with a force plot

In [5]:

```
shap.force_plot(explainer.expected_value[0], shap_values[0][0,:])
```

Out[5]:



Applying the SHAP for encrypted model. NTRU PQC encryption is used.

In [6]:

```
NTRU_ciphers = {0: 4848972778926144894, 1: 4848972778926144895, 2: 48489
import numpy as np
def replace_values_with_dict(arr, value_dict):
    # Use np.vectorize to apply the replacement function element-wise
    replace_func = np.vectorize(lambda x: value_dict.get(x, x))

    # Apply the replacement function to the entire array
    modified_array = replace_func(arr)

    return modified_array

X_test_ciphers = replace_values_with_dict(X_test.values, NTRU_ciphers)
```

In [11]:
```
# Generating encryopted model, i.e encrypting the thresholds.
import pdb
def replace_thresholds(tree, threshold_dict):
    """
    Replace decision thresholds in a scikit-learn DecisionTreeClassifier

    Parameters:
    - tree: The decision tree to be modified.
    - threshold_dict: A dictionary containing replacements for specific t

    Returns:
    - Modified decision tree.
    """
    if not hasattr(tree, 'tree_'):
        raise ValueError("Input must be a decision tree classifier.")

    # Retrieve decision thresholds from the tree
    thresholds = tree.tree_.threshold
    thresholds = [int(i) for i in thresholds]
    # Replace thresholds based on the dictionary

    th_cipher = []
    for value in thresholds:
        th_cipher.append(threshold_dict[value])

    for i in range(len(th_cipher)):
        tree.tree_.threshold[i] = th_cipher[i]

    return tree

clf_encrypted = replace_thresholds(clf, NTRU_ciphers)
```

Displaying the threshold values of unecryoted and encrypted model to show encryption.

In [14]:
```
loaded_clf = pickle.loads(saved_clf)
print(loaded_clf.tree_.threshold)
print(clf_encrypted.tree_.threshold)

explainer = shap.Explainer(clf)
shap_values = explainer.shap_values(X_test)
```

```
[126.5   0.5   1.5 ...  28.   -2.   -2. ]
[4.84925536e+18 4.84897278e+18 4.84897278e+18 ... 4.84897278e+18
 4.84897278e+18 4.84897278e+18]
```

Visualize the first 500 prediction's explanation with a force plot.

In [15]:
```
shap.force_plot(explainer.expected_value[0], shap_values[0][:500,:])
```

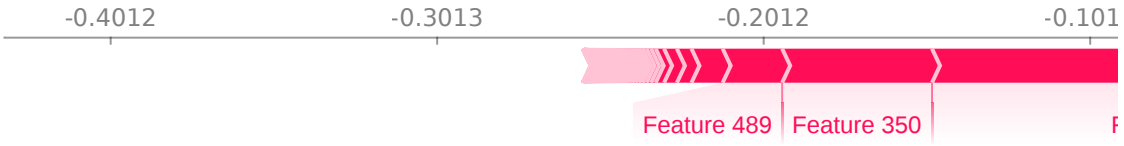
Out[15]:

Visualize the first prediction's explanation with a force plot.

In [16]:

```
shap.force_plot(explainer.expected_value[0], shap_values[0][0,:])
```

Out[16]:



In [ ]: