In [1]:
```
from sklearn.datasets import fetch_openml
from sklearn.metrics import accuracy_score,classification_report
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn import svm
```

In [16]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [8]:
```
mnist = fetch_openml('mnist_784')
```

```
C:\Users\LENOVO\AppData\Local\anaconda3\Lib\site-packages\sklearn\datasets\_openml.py:1002: FutureWarning
: The default value of `parser` will change from `'liac-arff'` to `'auto'` in 1.4. You can set `parser='a
uto'` to silence this warning. Therefore, an `ImportError` will be raised from 1.4 if the dataset is dens
e and pandas is not installed. Note that the pandas parser may return different data types. See the Notes
 Section in fetch_openml's API doc for details.  warn(
```

In [11]:
```
mnist.data.head
```

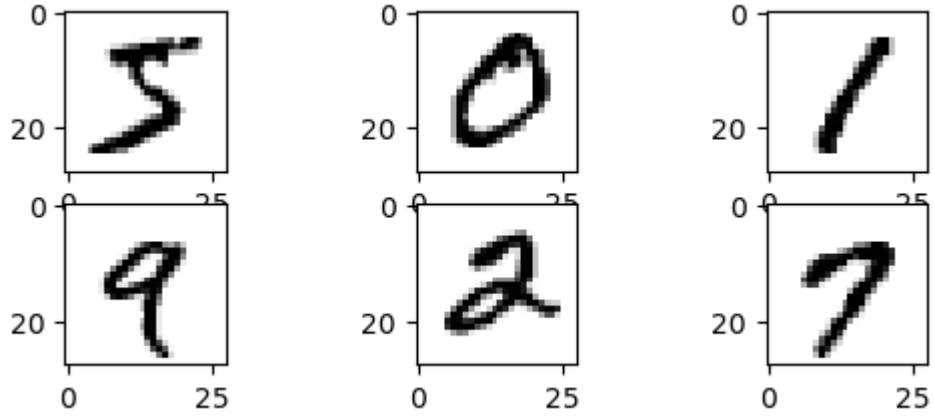
Out[11]:
```
<bound method NDFrame.head of        pixel1  pixel2  pixel3  pixel4  pixel5  pixel6  pixel7  pix
el8  pixel9  \0         0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   1
        0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   2         0.0
0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   3         0.0     0.0     0.0
0.0     0.0     0.0     0.0     0.0     0.0   4         0.0     0.0     0.0     0.0     0.0
0.0     0.0     0.0     0.0   ...       ...     ...     ...     ...     ...     ...     ...
...     ...   69995     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   69
996     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   69997     0.0
0.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0   69998     0.0     0.0     0.0
0.0     0.0     0.0     0.0     0.0     0.0   69999     0.0     0.0     0.0     0.0     0.0
0.0     0.0     0.0     0.0           pixel10  ...  pixel775  pixel776  pixel777  pixel778  pixel
779  \0          0.0  ...       0.0       0.0       0.0       0.0       0.0   1          0.0  ..
.       0.0       0.0       0.0       0.0       0.0   2          0.0  ...       0.0       0.0
    0.0       0.0       0.0   3          0.0  ...       0.0       0.0       0.0       0.0
0.0   4          0.0  ...       0.0       0.0       0.0       0.0       0.0   ...         ...  ..
.       ...       ...       ...       ...       ...   69995      0.0  ...       0.0       0.0
    0.0       0.0       0.0   69996      0.0  ...       0.0       0.0       0.0       0.0
0.0   69997      0.0  ...       0.0       0.0       0.0       0.0       0.0   69998      0.0  ..
.       0.0       0.0       0.0       0.0       0.0   69999      0.0  ...       0.0       0.0
    0.0       0.0       0.0           pixel780  pixel781  pixel782  pixel783  pixel784  0
   0.0       0.0       0.0       0.0       0.0  1           0.0       0.0       0.0       0.0
    0.0  2           0.0       0.0       0.0       0.0       0.0  3           0.0       0.0
  0.0       0.0       0.0  4           0.0       0.0       0.0       0.0       0.0  ...
...       ...       ...       ...       ...  69995       0.0       0.0       0.0       0.0
 0.0  69996       0.0       0.0       0.0       0.0       0.0  69997       0.0       0.0       0
.0       0.0       0.0  69998       0.0       0.0       0.0       0.0       0.0  69999       0.0
       0.0       0.0       0.0       0.0  [70000 rows x 784 columns]>
```

In [12]:
```
mnist.target.shape
```

Out[12]:
```
(70000,)
```

In [18]:
```
image= mnist.data.to_numpy()
plt.subplot(431)
plt.imshow((image[0].reshape(28,28)), cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(432)
plt.imshow(image[1].reshape(28,28), cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(433)
plt.imshow(image[3].reshape(28,28), cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(434)
plt.imshow(image[4].reshape(28,28), cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(435)
plt.imshow(image[5].reshape(28,28), cmap=plt.cm.gray_r,
interpolation='nearest')
plt.subplot(436)
plt.imshow(image[15].reshape(28,28), cmap=plt.cm.gray_r,
interpolation='nearest')
```

Out[18]:
```
<matplotlib.image.AxesImage at 0x1c95faafed0>
```



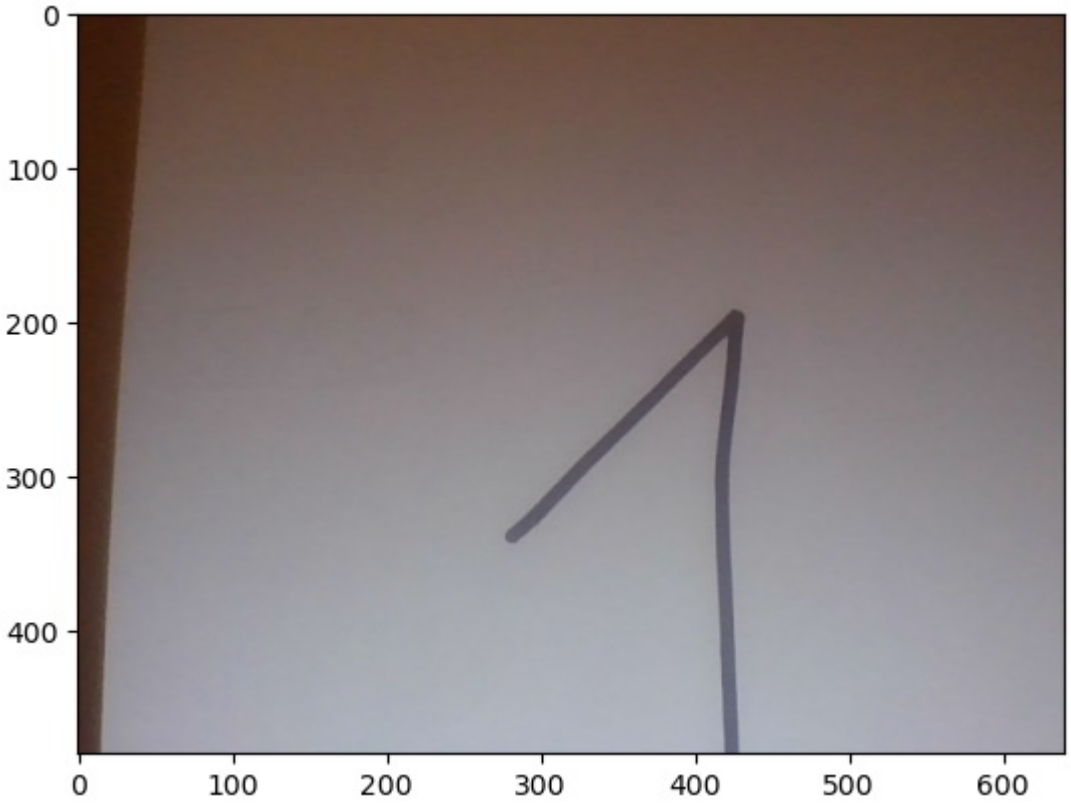
In [19]:
```
index_number= np.random.permutation(70000)

x1,y1=mnist.data.loc[index_number],mnist.target.loc[index_number]
x1.reset_index(drop=True,inplace=True)
y1.reset_index(drop=True,inplace=True)

x_train , x_test = x1[:55000], x1[55000:]

y_train , y_test = y1[:55000], y1[55000:]
```

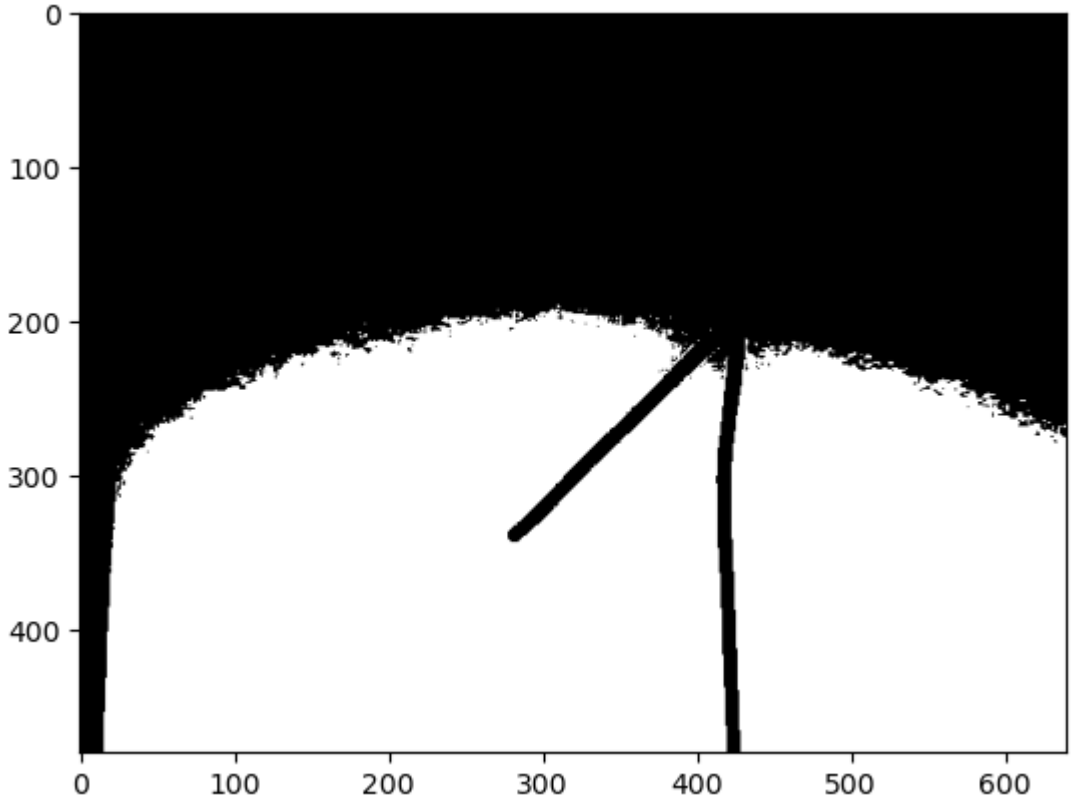
In [20]:
```
svc = svm.SVC(gamma='scale',class_weight='balanced',C=100)
svc.fit(x_train,y_train)
result=svc.predict(x_test)

print('Accuracy :',accuracy_score(y_test,result))
print(classification_report(y_test,result))
```

```
Accuracy : 0.9838              precision    recall  f1-score   support           0       0.99      0.99
    0.99      1462           1       0.99      0.99      0.99      1662           2       0.98      0.98
     0.98      1537           3       0.98      0.98      0.98      1463           4       0.98      0.98
      0.98      1471           5       0.98      0.98      0.98      1349           6       0.98      0.9
9      0.99      1537           7       0.98      0.98      0.98      1595           8       0.98      0.
98      0.98      1455           9       0.98      0.97      0.98      1469    accuracy
        0.98     15000   macro avg       0.98      0.98      0.98     15000weighted avg       0.98
0.98      0.98     15000
```

In [75]:
```
from PIL import Image
import cv2
import numpy as np
import matplotlib.pyplot as plt


def capture_image():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    if ret:
        plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        plt.show()
    else:
        print("Failed to capture image")
    cap.release()
    cv2.destroyAllWindows()
    return frame if ret else None


def pre_process_image(image):
    if image is not None:

        gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


        _, gray_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)


        plt.imshow(gray_image, cmap='gray')
        plt.show()


        resized_image = cv2.resize(gray_image, (28, 28), interpolation=cv2.INTER_AREA)
```

```
        normalized_image = resized_image.astype(np.float32) / 255.0


        flattened_image = normalized_image.flatten()

        return flattened_image
    else:
        return None



image = capture_image()
if image is not None:
    transformed_image = pre_process_image(image)
    if transformed_image is not None:
        prediction = svc.predict(transformed_image.reshape(1, -1))
        print(prediction)
    else:
        print("Error in image transformation")
else:

    print("No image captured")
```



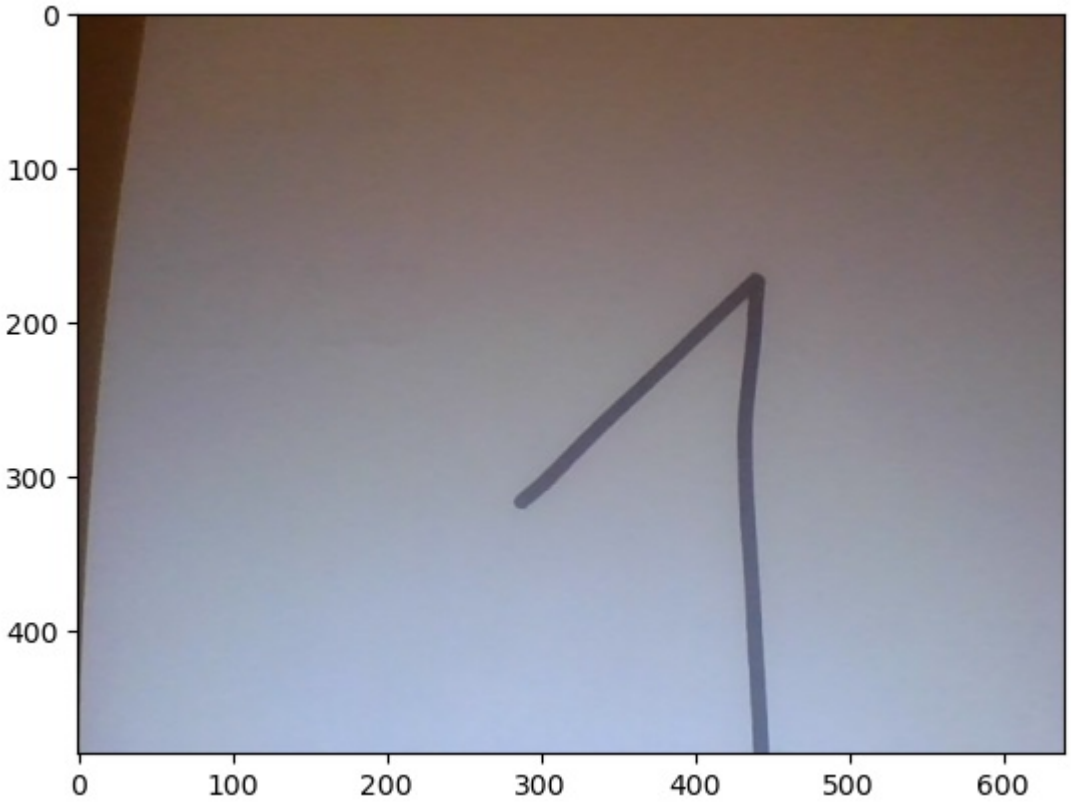


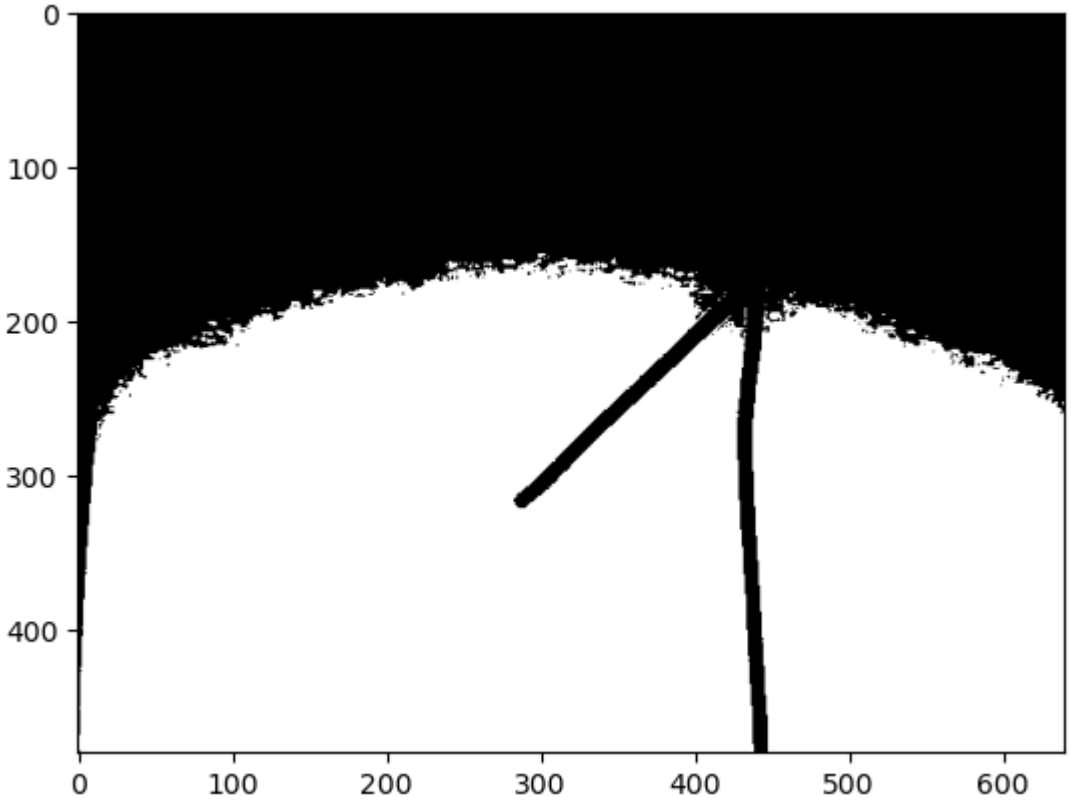

```
['1']
C:\Users\LENOVO\AppData\Local\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not ha
ve valid feature names, but SVC was fitted with feature names  warnings.warn(
```

In [69]:
```
knn = KNeighborsClassifier(n_neighbors=6,weights='distance')
knn.fit(x_train, y_train)

result=knn.predict(x_test)
print('Accuracy :',accuracy_score(y_test,result))
print(classification_report(y_test,result))
```

```
Accuracy : 0.9722              precision    recall  f1-score   support           0       0.98      0.99
    0.99      1462           1       0.96      1.00      0.98      1662           2       0.99      0.97
     0.98      1537           3       0.97      0.97      0.97      1463           4       0.98      0.97
      0.97      1471           5       0.97      0.97      0.97      1349           6       0.98      0.9
9      0.98      1537           7       0.96      0.97      0.97      1595           8       0.99      0.
93      0.96      1455           9       0.95      0.96      0.96      1469    accuracy
        0.97     15000   macro avg       0.97      0.97      0.97     15000weighted avg       0.97
0.97      0.97     15000
```

In [74]:
```
image = capture_image()


transformed_image = pre_process_image(image)



prediction = svc.predict(transformed_image.reshape(1, -1))

print(prediction)
```






```
['1']
C:\Users\LENOVO\AppData\Local\anaconda3\Lib\site-packages\sklearn\base.py:464: UserWarning: X does not ha
ve valid feature names, but SVC was fitted with feature names  warnings.warn(
```

In [76]:
```
!pip install Flask
```

```
Requirement already satisfied: Flask in c:\users\lenovo\appdata\local\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\lenovo\appdata\local\anaconda3\lib\site-packag
es (from Flask) (2.2.3)Requirement already satisfied: Jinja2>=3.0 in c:\users\lenovo\appdata\local\anacon
da3\lib\site-packages (from Flask) (3.1.2)Requirement already satisfied: itsdangerous>=2.0 in c:\users\le
novo\appdata\local\anaconda3\lib\site-packages (from Flask) (2.0.1)Requirement already satisfied: click>=
8.0 in c:\users\lenovo\appdata\local\anaconda3\lib\site-packages (from Flask) (8.0.4)Requirement already
satisfied: colorama in c:\users\lenovo\appdata\local\anaconda3\lib\site-packages (from click>=8.0->Flask)
 (0.4.6)Requirement already satisfied: MarkupSafe>=2.0 in c:\users\lenovo\appdata\local\anaconda3\lib\sit
e-packages (from Jinja2>=3.0->Flask) (2.1.1)
```

In [79]:
```
from flask import Flask, request, render_template_string
import cv2
import numpy as np
```

```
app = Flask(__name__)

UPLOAD_FORM = """
<!doctype html>
<title>Upload Image</title>
<h1>Upload Image</h1>
<form method=post enctype=multipart/form-data>
  <input type=file name=file>
  <input type=submit value=Upload>
</form>
"""


def pre_process_image(image):

    pass

@app.route('/', methods=['GET', 'POST'])
def upload_image():
    if request.method == 'POST':

        if 'file' not in request.files:
            return 'No file part'
        file = request.files['file']

        if file.filename == '':
            return 'No selected file'
        if file:

            in_memory_file = np.frombuffer(file.read(), np.uint8)
            image = cv2.imdecode(in_memory_file, cv2.IMREAD_COLOR)


            transformed_image = pre_process_image(image)


            prediction = svc.predict(transformed_image.reshape(1, -1))


            return f'Prediction: {prediction}'
    return render_template_string(UPLOAD_FORM)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```

```
 * Serving Flask app '__main__' * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI se
rver instead. * Running on all addresses (0.0.0.0) * Running on http://127.0.0.1:8000 * Running on http:/
/192.168.0.14:8000Press CTRL+C to quit * Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.SystemExit: 1
```

In [ ]:

In [ ]: