برای حل مسئله شناسایی ارقام دستنویس با استفاده از کتابخانه scikitlearn ابتدا باید کتابخانه های مورد نظر را import کنیم. کتابخانه هایی که در این پیاده سازی نیاز داریم عبارتند از:

1- numpy
2- matplotlib
3- scikitlearn

از کتابخانه scikitlearn پنج قسمت مورد نیاز را import می‌کنیم.

ابتدا با استفاده از fetch_openml دیتاست مورد نیاز خود را از آدرس https://www.openml.org/d/554 دانلود می‌کنیم. fetch_openml با استفاده از پارامتر های ورودی خود داده های را به صورتی که ما نیاز داریم parse کرده و در ۲ آرایه x و y به ما برمی‌گرداند.

در مرحله بعدی نیاز است که یک permutation از داده های x و y را به صورت رندوم بدست آوریم. در خطوط زیر این عملیات انجام می‌شود.

```
random_state = check_random_state(0)
permutation = random_state.permutation(X.shape[0])
X = X[permutation]
y = y[permutation]
X = X.reshape((X.shape[0], -1))
```

با استفاده از تابع train_test_split و با کمک پارامتر های ورودی دیتاست را به دو دسته test و train تقسیم می‌کنیم. ما اینجا از کل داده های ورودی ۱۰هزار داده با به عنوان داده تست در نظر می‌گیریم. این تابع داده های تست و آموزش را به صورت ۴ آرایه به ما برمی‌گرداند.

پس از تقسیم بعدی داده های به دو دسته تست و آموزش داده ها را استاندارد سازی می‌کنیم. با استفاده از تابع StandardScaler میانگین ها را حذف کرده و داده ها را به unit variance تغییر می‌دهیم.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

برای پوشش بیشتر با استفاده از تابع LogesticRegression میزان تحمل یا tolerance داده های آموزشی را افزایش می‌دهم.

```
clf = LogisticRegression(C=50.0 / train_samples,
penalty="l1", solver="saga", tol=0.1)
clf.fit(X_train, y_train)
```

در نهایت میزان sparsity و score را روی داده های تست محاسبه کرده تا میزان دقت الگوریتم را بدست بیاوریم.

```
sparsity = np.mean(clf.coef_ == 0) * 100
score = clf.score(X_test, y_test)
```

در انتها با استفاده از یک حلقه و کتابخانه matplotlib یک نمای کلی از نتیجه الگوریتم را نمایش می‌دهیم.