In [23]:

```
import numpy as np
import pandas as pd
```

In [24]:

```
df = pd.read_csv('train.csv')
```

In [25]:

```
df.shape
```

Out[25]:

```
(42000, 785)
```

In [26]:

```
df.sample()
```

Out[26]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774
5122	8	0	0	0	0	0	0	0	0	0	...	0

1 rows × 785 columns

In [5]:

```
import matplotlib.pyplot as plt
```

In [28]:

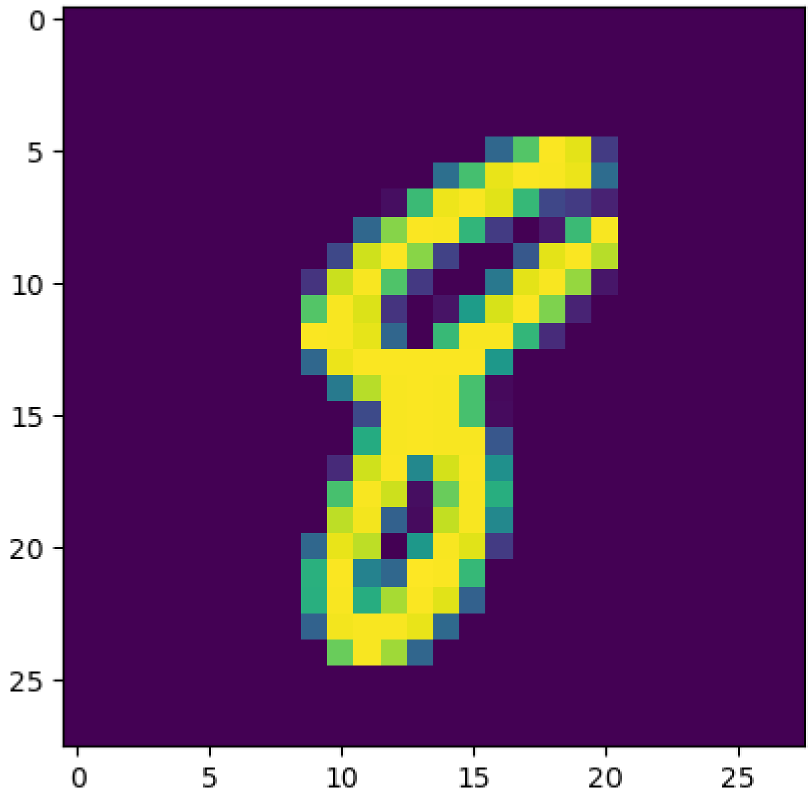
```
#df.iloc[35583,1:].values.reshape(28,28)
```

In [29]:

```
plt.imshow(df.iloc[5122,1:].values.reshape(28,28))
```

Out[29]:

```
<matplotlib.image.AxesImage at 0x23cf8874f70>
```



In [30]:

```
X = df.iloc[:,1:]
y = df.iloc[:,0]
```

In [31]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [32]:

```
X_train.shape
```

Out[32]:

```
(33600, 784)
```

In [33]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

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

In [34]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [35]:

```
knn = KNeighborsClassifier()
```

In [36]:

```
knn.fit(X_train,y_train)
```

Out[36]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [37]:

```
y_pred = knn.predict(X_test)
```

In [38]:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[38]:

```
0.9391666666666667
```

In [39]:

```
# PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=200)
```

In [40]:

```
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.transform(X_test)
```

In [41]:

```
X_train_trf.shape
```

Out[41]:

```
(33600, 200)
```

In [42]:

```
knn = KNeighborsClassifier()
```

In [43]:

```
knn.fit(X_train_trf,y_train)
```

Out[43]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [44]:

```
y_pred = knn.predict(X_test_trf)
```

In [45]:

```
accuracy_score(y_test,y_pred)
```

Out[45]:

```
0.9502380952380952
```

In [22]:

```
for i in range(1,785):
    pca = PCA(n_components=i)
    X_train_trf = pca.fit_transform(X_train)
    X_test_trf = pca.transform(X_test)

    knn = KNeighborsClassifier()

    knn.fit(X_train_trf,y_train)

    y_pred = knn.predict(X_test_trf)

    print(accuracy_score(y_test,y_pred))

```

```
0.27166666666666667
0.42238095238095236
0.4851190476190476
0.6194047619047619
0.7297619047619047
0.8188095238095238


---------------------------------------------------------------------------
-
KeyboardInterrupt                          Traceback (most recent call las
t)
Cell In[22], line 8
      4 X_test_trf = pca.transform(X_test)
      6 knn = KNeighborsClassifier()
----> 8 knn.fit(X_train_trf,y_train)
     10 y_pred = knn.predict(X_test_trf)
     12 print(accuracy_score(y_test,y_pred))

File ~\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:21
5, in KNeighborsClassifier.fit(self, X, y)
    196 """Fit the k-nearest neighbors classifier from the training datase
t.
    197
    198 Parameters
   (...)
    211     The fitted k-nearest neighbors classifier.
    212 """
    213 self._validate_params()
--> 215 return self._fit(X, y)

KeyboardInterrupt: 
```

In [46]:

```
# transforming to a 2D coordinate system
pca = PCA(n_components=2)
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.transform(X_test)
```

In [47]:

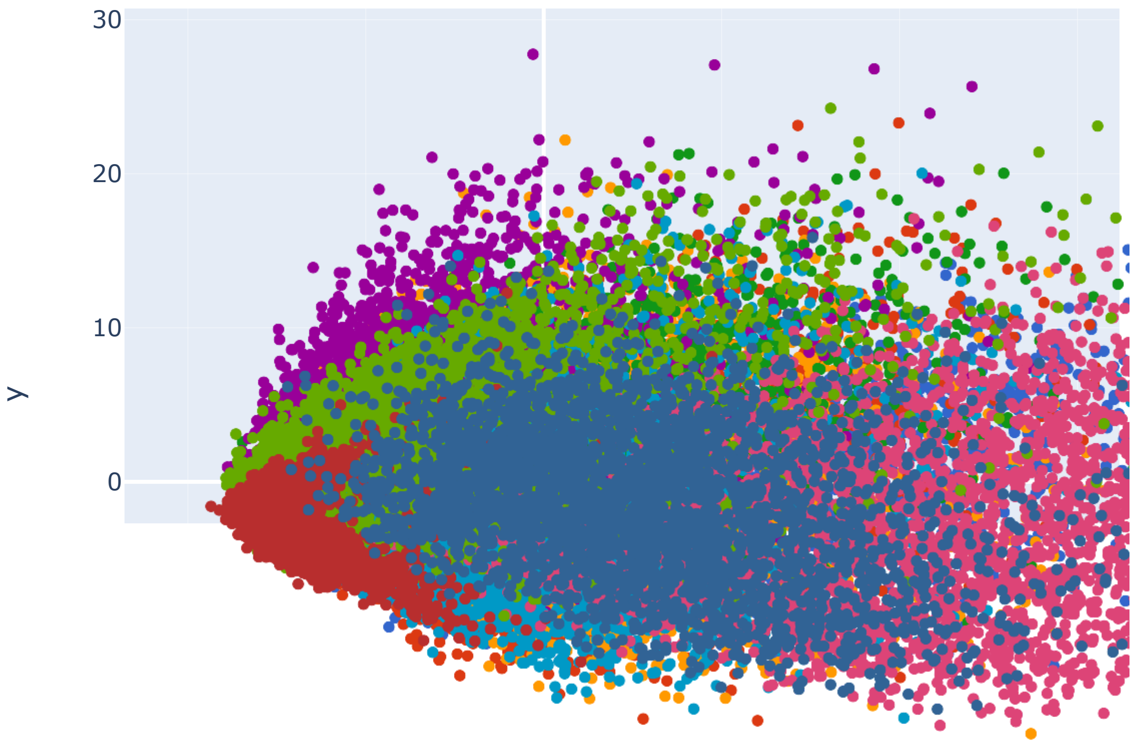
```
X_train_trf
```

Out[47]:

```
array([[-2.71862909, -0.49008924],
       [-0.67695432, -6.7532776 ],
       [-3.0332419 ,  6.51020247],
       ...,
       [ 2.14883801,  0.78058549],
       [ 1.05956236,  0.94664421],
       [17.70259744,  1.96015569]])
```

In [52]:

```
import plotly.express as px
y_train_trf = y_train.astype(str)
fig = px.scatter(x=X_train_trf[:,0],
                 y=X_train_trf[:,1],
                 color=y_train_trf,
                 color_discrete_sequence=px.colors.qualitative.G10
                )
fig.show()
```



In [53]:

```python
# transforming in 3D
pca = PCA(n_components=3)
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.transform(X_test)
```

In [54]:

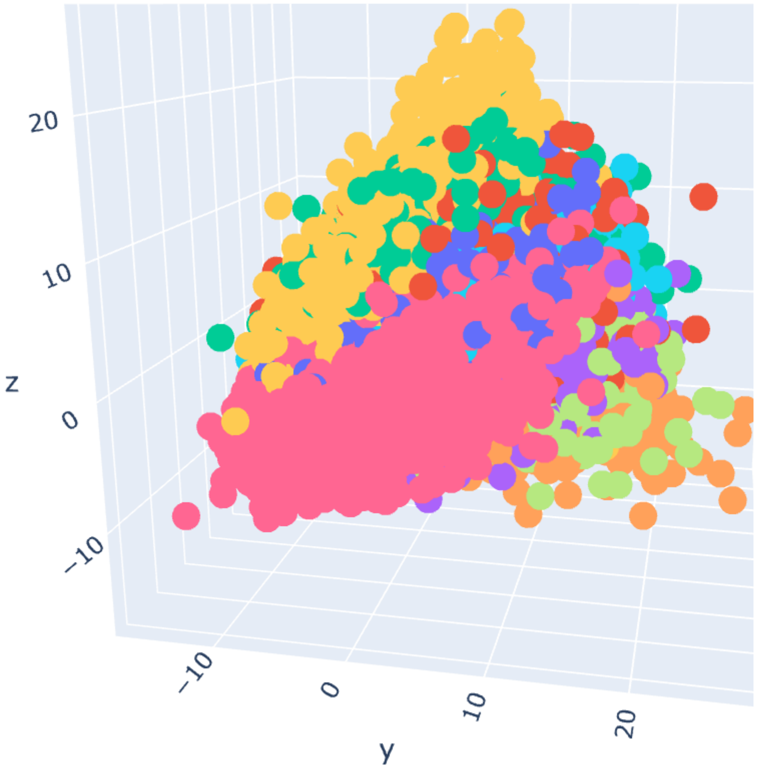
```python
X_train_trf
```

Out[54]:

```
array([[-2.71862824, -0.48998695,  1.13553626],
       [-0.67698699, -6.7531474 , -2.33606672],
       [-3.03323282,  6.50915021,  7.49143667],
       ...,
       [ 2.14882549,  0.78131841, -0.74665575],
       [ 1.05958282,  0.94717495,  3.94984106],
       [17.70259048,  1.96129657, -4.94409932]])
```

In [55]:

```
import plotly.express as px
y_train_trf = y_train.astype(str)
fig = px.scatter_3d(df, x=X_train_trf[:,0], y=X_train_trf[:,1], z=X_train_trf[:,2],
              color=y_train_trf)
fig.update_layout(
    margin=dict(l=20, r=20, t=20, b=20),
    paper_bgcolor="LightSteelBlue",
)
fig.show()
```



In [56]:

```
pca.explained_variance_
# Eigen values
```

Out[56]:

```
array([40.67111198, 29.17023382, 26.74459606])
```

In [57]:

```
pca.components_.shape
# Eigen vectors
```

Out[57]:

```
(3, 784)
```

In [58]:

```
pca.explained_variance_ratio_
```

Out[58]:

```
array([0.05785192, 0.0414927 , 0.03804239])
```

In [59]:

```
pca = PCA(n_components=None)
X_train_trf = pca.fit_transform(X_train)
X_test_trf = pca.transform(X_test)
```

In [60]:

```
pca.explained_variance_.shape
```

Out[60]:

```
(784,)
```

In [61]:

```
pca.components_.shape
```

Out[61]:

```
(784, 784)
```

In [70]:

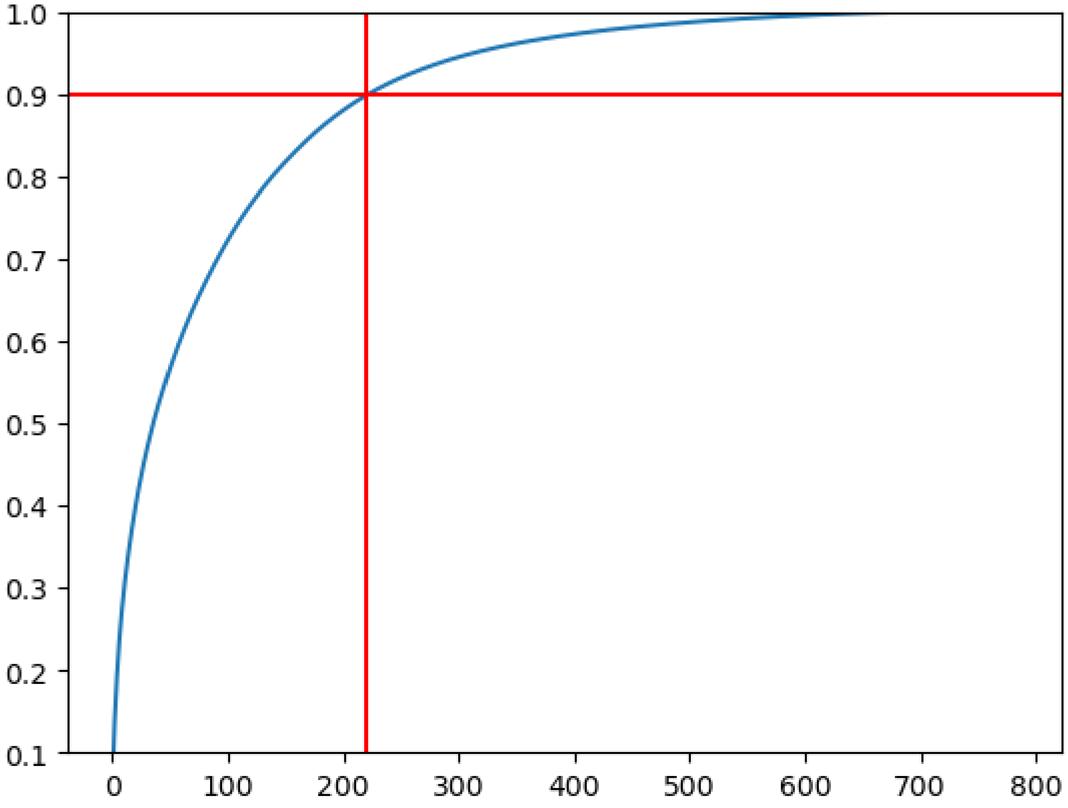
```
np.cumsum(pca.explained_variance_ratio_)[:10]
```

Out[70]:

```
array([0.05785192, 0.09934462, 0.13738701, 0.16704964, 0.19286525,
       0.21541506, 0.23514574, 0.25289854, 0.26858504, 0.28294568])
```

In [69]:

```
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.ylim((0.1,1))
plt.axhline(0.9, color = "red")
plt.axvline(220, color = "red")
plt.show()
```



In [ ]:

```

```