In [ ]:

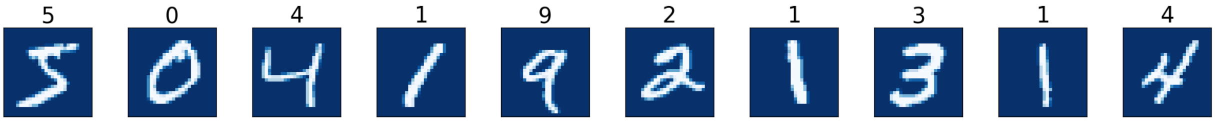
```
from sklearn.preprocessing import scale
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
%matplotlib inline
```

In [ ]:

```
from keras.datasets import mnist
(train_x, train_y), (test_x, test_y) = mnist.load_data()
```

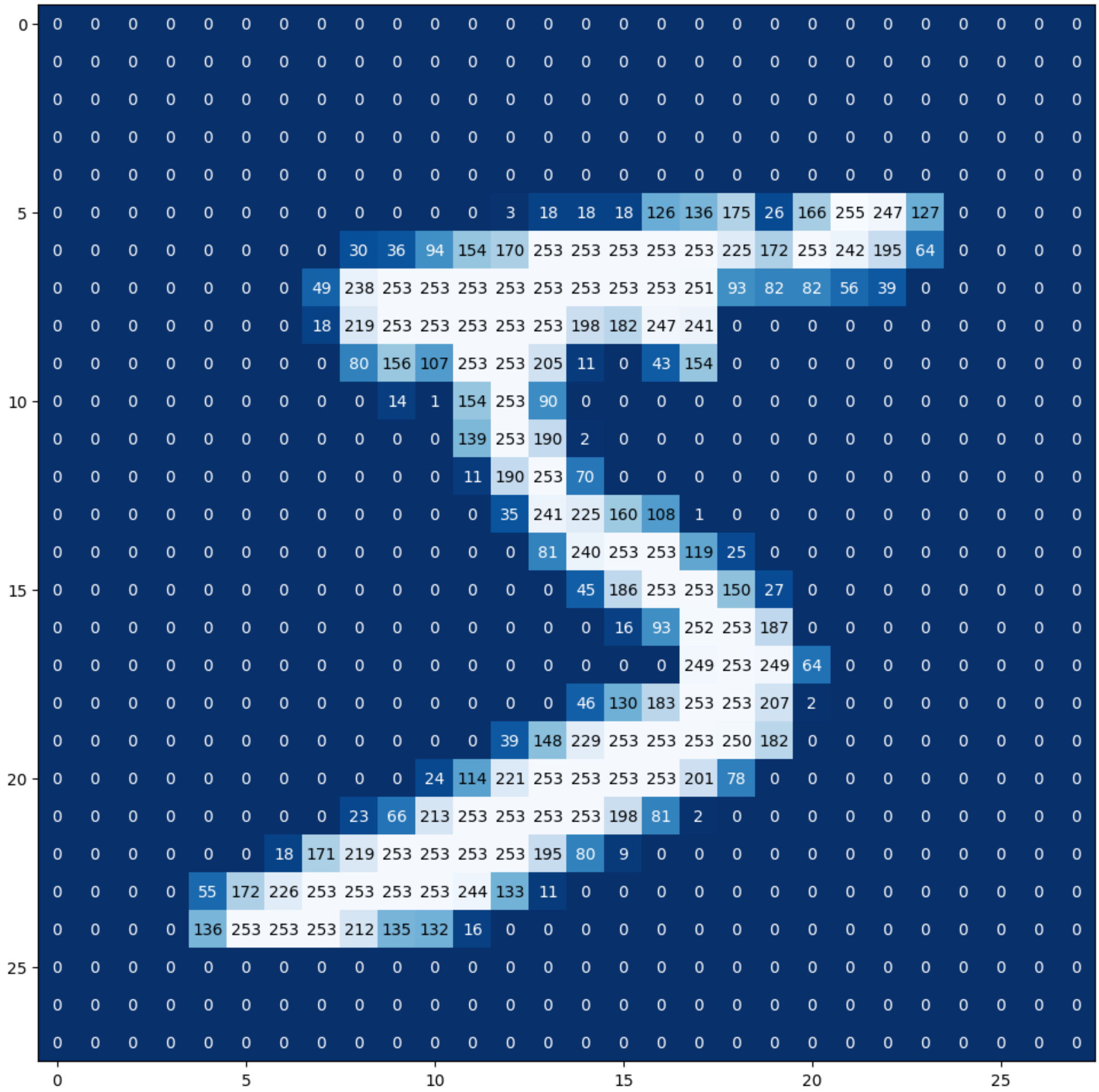
In [ ]:

```
fig = plt.figure(figsize=(25, 4))
for idx in np.arange(10):
    ax = fig.add_subplot(2, 10, idx+1, xticks=[], yticks=[])
    ax.imshow(train_x[idx], cmap='Blues_r')
    ax.set_title(str(train_y[idx]),fontsize=25)
```



In [ ]:

```
img = train_x[0]

fig = plt.figure(figsize = (12,12))
ax = fig.add_subplot(111)
ax.imshow(img, cmap='Blues_r')
width, height = img.shape
thresh = img.max()/2.5
for x in range(width):
    for y in range(height):
        val = round(img[x][y],2) if img[x][y] !=0 else 0
        ax.annotate(str(val), xy=(y,x),
                    horizontalalignment='center',
                    verticalalignment='center',
                    color='white' if img[x][y]<thresh else 'black')
```



```
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136 175  26 166 255 247 127   0   0   0   0
  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253 225 172 253 242 195  64   0   0   0   0
  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251  93  82  82  56  39   0   0   0   0   0
  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119  25   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253 150  27   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252 253 187   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249 253 249  64   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253 253 207   2   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253 250 182   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201  78   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

In [ ]:

```
train_x = train_x.reshape(train_x.shape[0], -1)
print(train_x.shape)

sample_size = 5000
# Use only the top 1000 data for training
train_x = pd.DataFrame(train_x[:sample_size, :])
train_y = train_y[:sample_size]
```

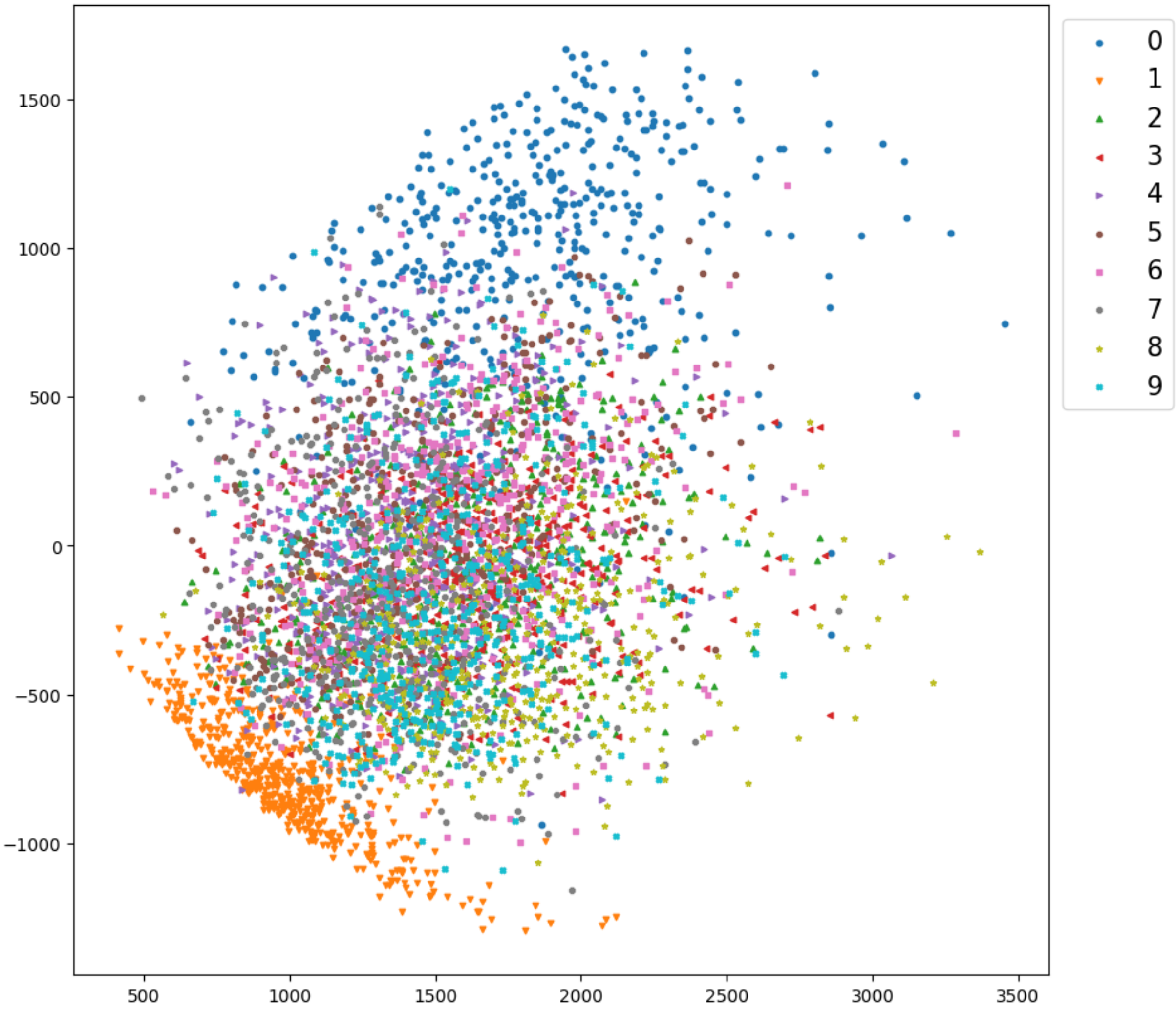
```
(60000, 784)
```

In [ ]:

```
from sklearn.decomposition import TruncatedSVD

tsvd = TruncatedSVD()
x_tsvd = tsvd.fit_transform(train_x)
markers=['o','v','^','<','>','8','s','P','*','X']
# plot in 2D by class
plt.figure(figsize=(10,10))
for i,marker in enumerate(markers):
    mask = train_y == i
    plt.scatter(x_tsvd[mask, 0], x_tsvd[mask, 1], label=i, s=10, alpha=1,marker=marke
plt.legend(bbox_to_anchor=(1.00, 1), loc='upper left',fontsize=15)
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x261b799a340>
```

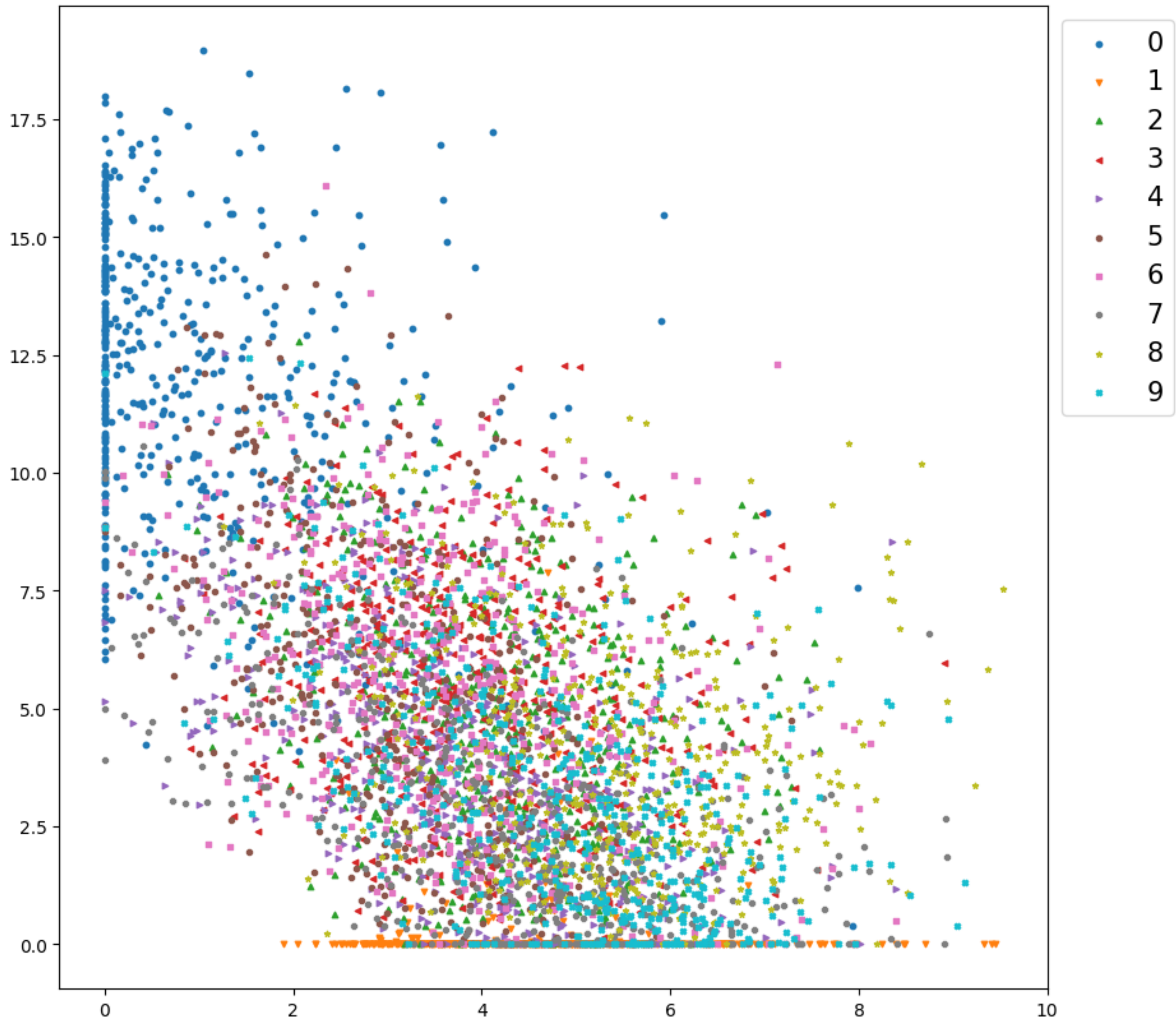


In [ ]:

```
from sklearn.decomposition import NMF

nmf = NMF(n_components=2, init='random', random_state=0)
x_nmf = nmf.fit_transform(train_x)
markers=['o','v','^','<','>','8','s','P','*','X']
# plot in 2D by class
plt.figure(figsize=(10,10))
for i,marker in enumerate(markers):
    mask = train_y == i
    plt.scatter(x_nmf[mask, 0], x_nmf[mask, 1], label=i, s=10, alpha=1,marker=marker)
plt.legend(bbox_to_anchor=(1.00, 1), loc='upper left',fontsize=15)
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x261b78c20a0>
```

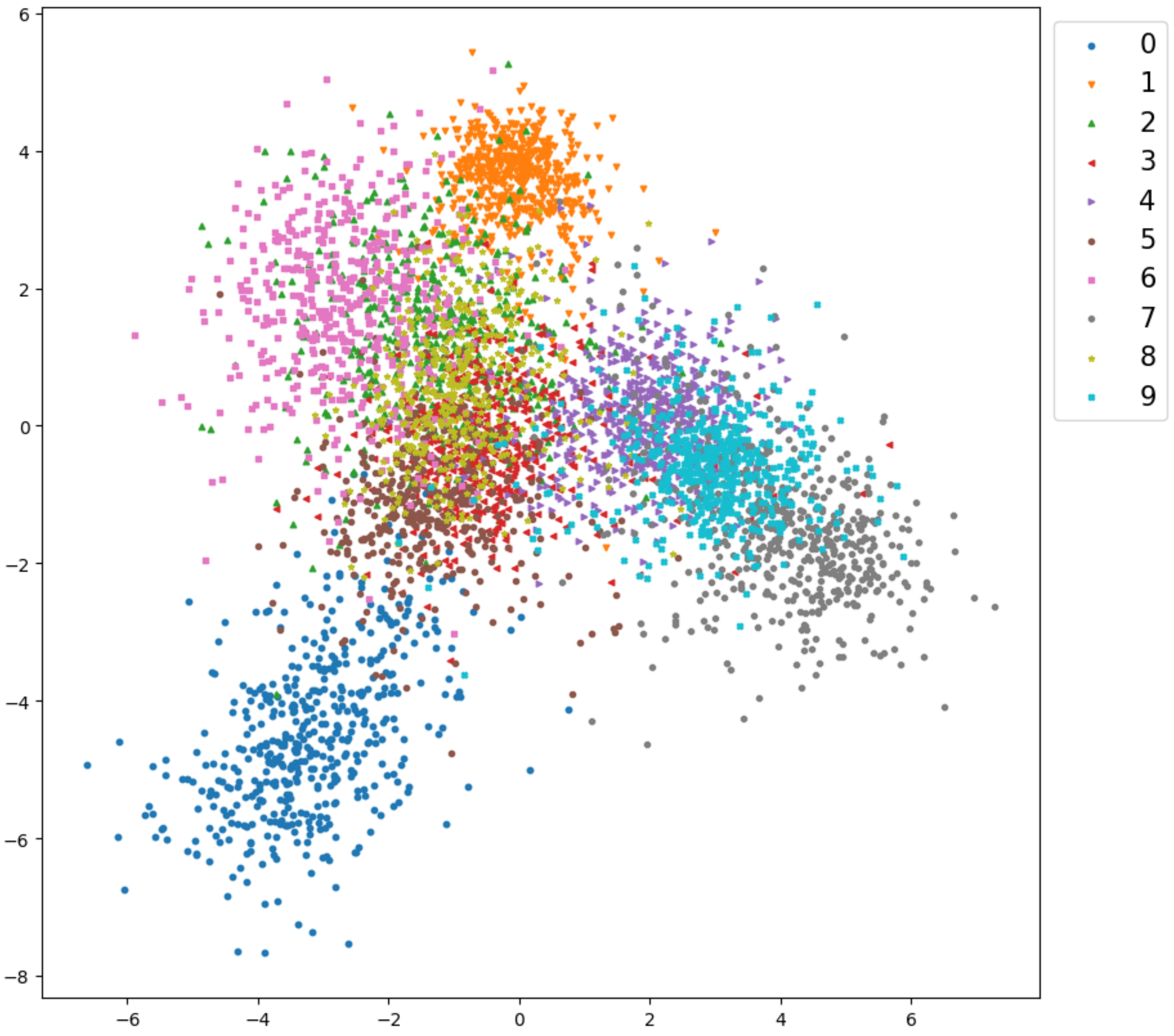


In [ ]:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

lda = LDA(n_components=2)
x_lda = lda.fit_transform(train_x, train_y)

plt.figure(figsize=(10,10))
for i,marker in enumerate(markers):
    mask = train_y == i
    plt.scatter(x_lda[mask, 0], x_lda[mask, 1], label=i, s=10, alpha=1,marker=marker)
plt.legend(bbox_to_anchor=(1.00, 1), loc='upper left',fontsize=15)
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x2619c4e8580>
```

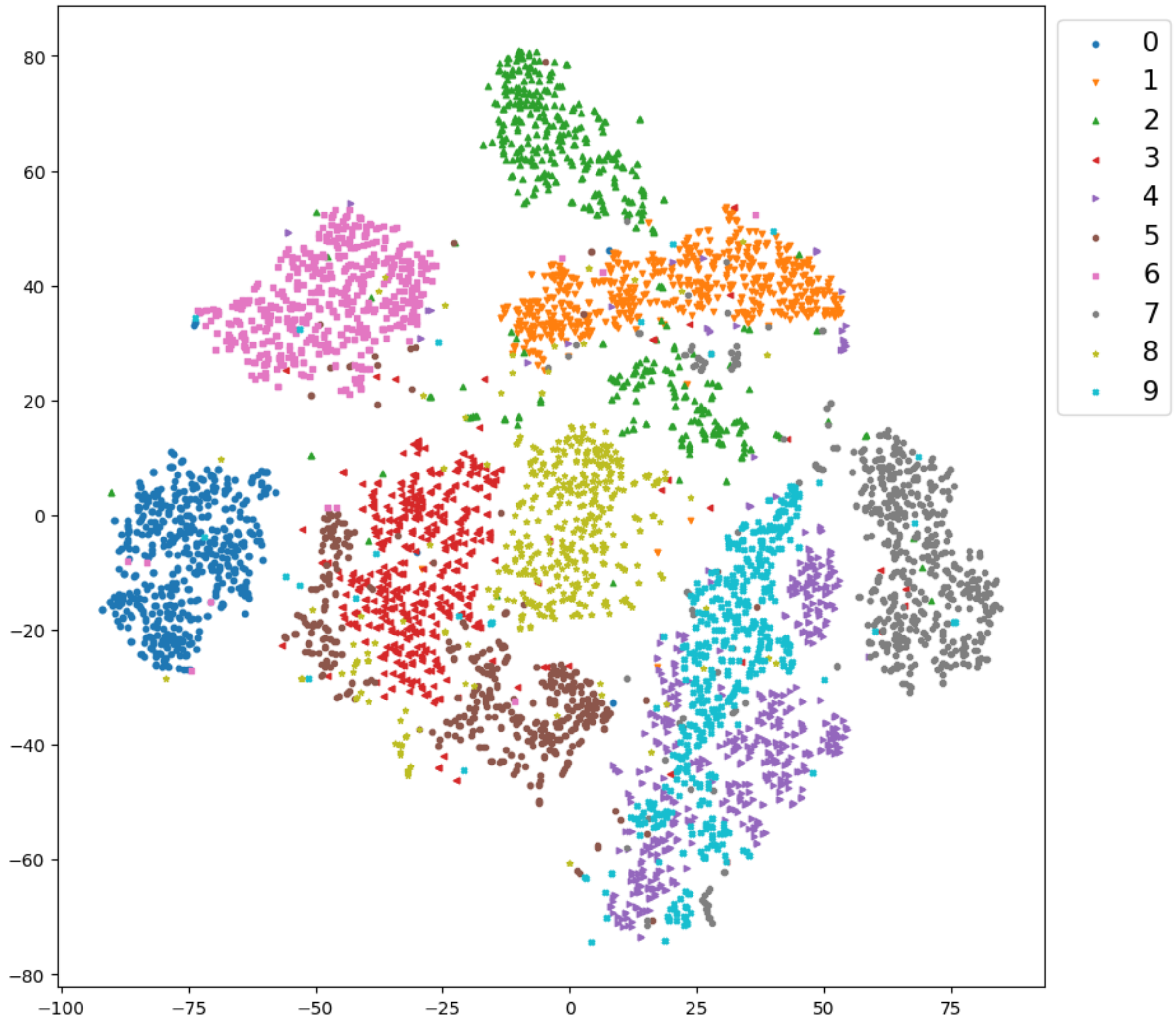


In [ ]:

```
from sklearn.manifold import TSNE

tsne = TSNE(n_components=2)
x_tsne = tsne.fit_transform(train_x)

plt.figure(figsize=(10,10))
for  i,marker  in  enumerate(markers):
    mask = train_y == i
    plt.scatter(x_tsne[mask, 0], x_tsne[mask, 1], label=i, s=10, alpha=1,marker=marke
plt.legend(bbox_to_anchor=(1.00, 1), loc='upper left',fontsize=15)
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x261b5d9e6d0>
```



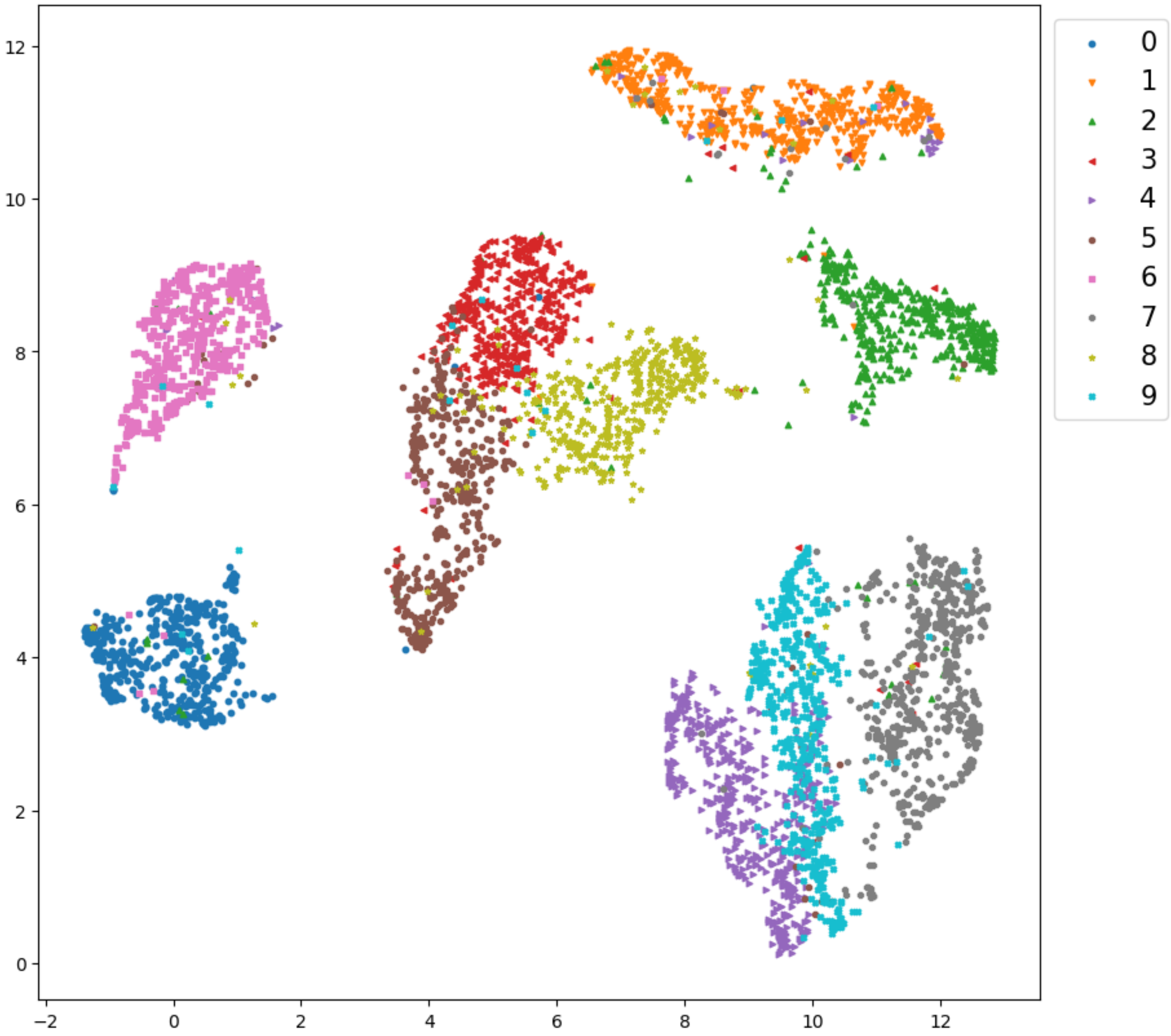
In [ ]:

```
import umap

um = umap.UMAP()
x_umap = um.fit_transform(train_x)

plt.figure(figsize=(10,10))
for i,marker in enumerate(markers):
    mask = train_y == i
    plt.scatter(x_umap[mask, 0], x_umap[mask, 1], label=i, s=10, alpha=1,marker=marke
plt.legend(bbox_to_anchor=(1.00, 1), loc='upper left',fontsize=15)
```

```
c:₩ProgramData₩Anaconda3₩envs₩yujinyeo₩lib₩site-packages₩numba₩core₩cpu.py:97: UserWar
ning: Numba extension module 'numba_dppy.numpy_usm_shared' failed to load due to 'Impo
rtError(cannot import name 'TargetConfig' from 'numba.core.dispatcher' (c:₩ProgramData
₩Anaconda3₩envs₩yujinyeo₩lib₩site-packages₩numba₩core₩dispatcher.py))'.
  numba.core.entrypoints.init_all()
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x261dd3ba850>
```



In [ ]:

```

```

In [ ]:

```
train = pd.read_csv(r"C:₩temp₩train.csv")

target = train['target']
train = train.drop(["target", "ID"], axis=1)

print ("Rows: " + str(train.shape[0]) + ", Columns: " + str(train.shape[1]))
train.head()
```

```
Rows: 4459, Columns: 4991
```

Out[ ]:

	48df886f9	0deb4b6a8	34b15f335	a8cb14b00	2f0771a37	30347e683	d08d1fbe3	6ee66e115
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

5 rows × 4991 columns

In [ ]:

```
from sklearn.preprocessing import StandardScaler
```

```
standardized_train = StandardScaler().fit_transform(train.values)
```

In [ ]:

```
import chart_studio.plotly as py
import cufflinks as cf


import plotly.graph_objs as go
from plotly.offline import iplot

cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
```

In [ ]:

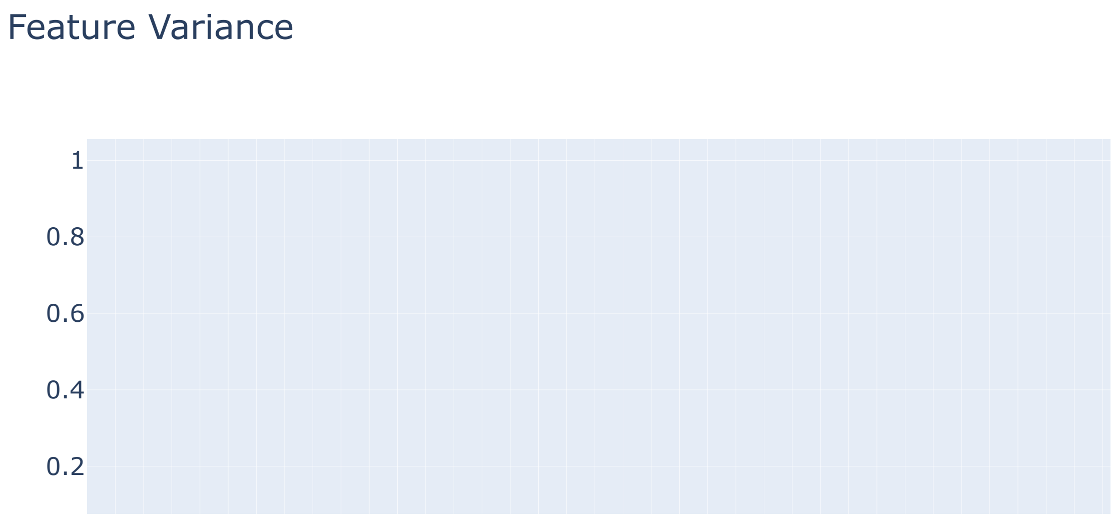
```
feature_df = train.describe().T
feature_df = feature_df.reset_index().rename(columns = {'index' : 'columns'})
feature_df['distinct_vals'] = feature_df['columns'].apply(lambda x : len(train[x].va
feature_df['column_var'] = feature_df['columns'].apply(lambda x : np.var(train[x]))
feature_df['column_std'] = feature_df['columns'].apply(lambda x : np.std(train[x]))
feature_df['column_mean'] = feature_df['columns'].apply(lambda x : np.mean(train[x])
feature_df['target_corr'] = feature_df['columns'].apply(lambda x : np.corrcoef(targe
feature_df.head()
```

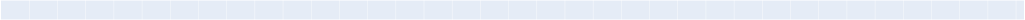
Out[ ]:

	columns	count	mean	std	min	25%	50%	75%	max	distinct_vals
0	48df886f9	4459.0	14654.930101	3.893298e+05	0.0	0.0	0.0	0.0	20000000.0	32
1	0deb4b6a8	4459.0	1390.894819	6.428302e+04	0.0	0.0	0.0	0.0	4000000.0	5
2	34b15f335	4459.0	26722.450922	5.699652e+05	0.0	0.0	0.0	0.0	20000000.0	29
3	a8cb14b00	4459.0	4530.163714	2.359124e+05	0.0	0.0	0.0	0.0	14800000.0	3
4	2f0771a37	4459.0	26409.957390	1.514730e+06	0.0	0.0	0.0	0.0	100000000.0	6

In [ ]:

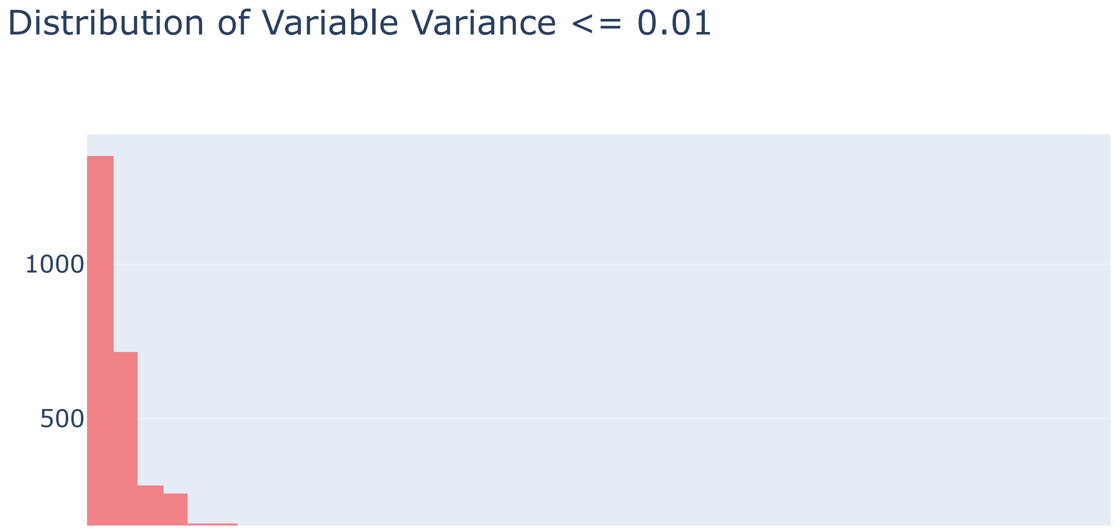
```
feature_df = feature_df.sort_values('column_var', ascending = True)
feature_df['column_var'] = (feature_df['column_var'] - feature_df['column_var'].min()
trace1 = go.Scatter(x=feature_df['columns'], y=feature_df['column_var'], opacity=0.75
layout = dict(height=400, title='Feature Variance', legend=dict(orientation="h"));
fig = go.Figure(data=[trace1], layout=layout);
iplot(fig);
```

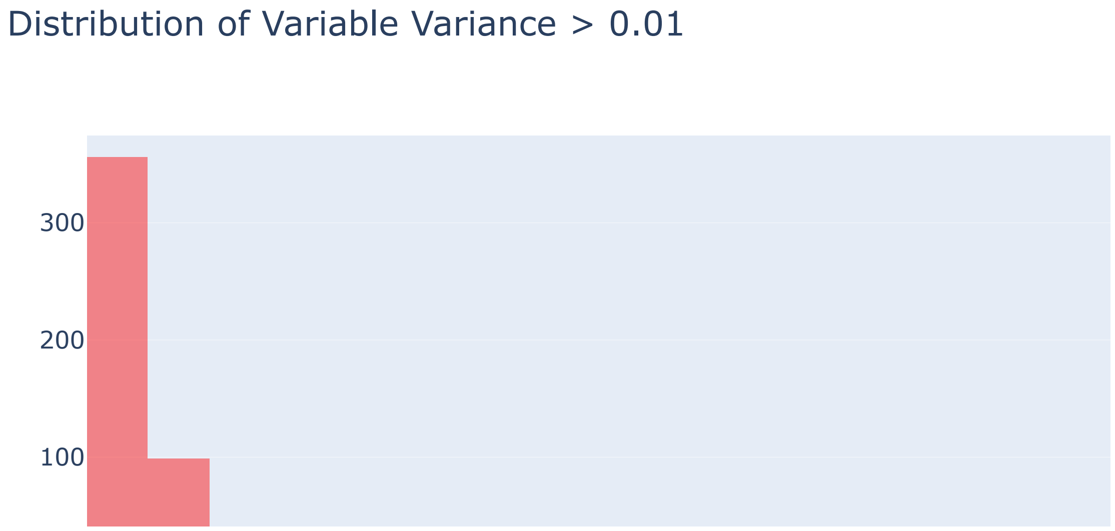




In [ ]:

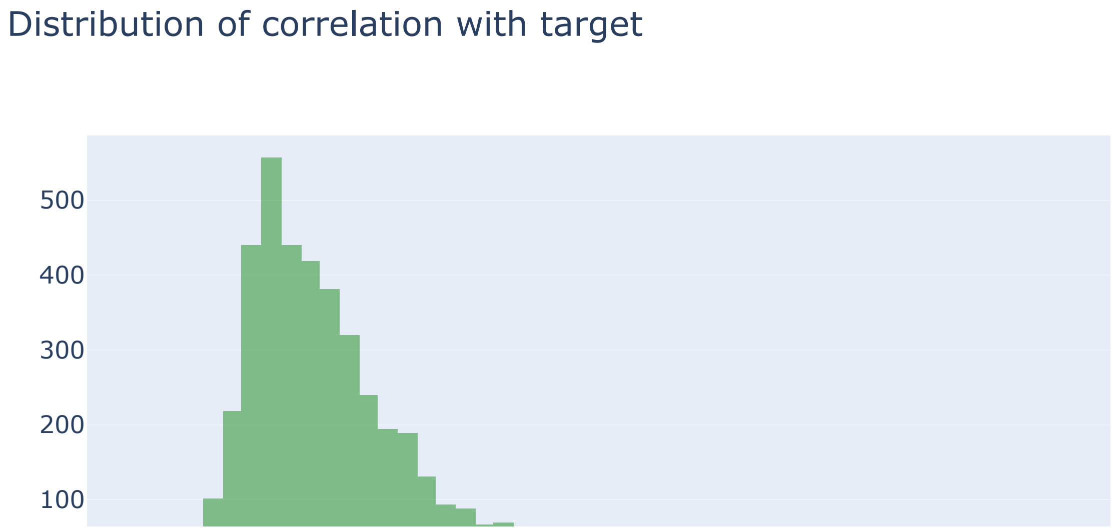
```
trace1 = go.Histogram(x=feature_df[feature_df['column_var'] <= 0.01]['column_var'], o
layout = dict(height=400, title='Distribution of Variable Variance <= 0.01', legend=d
fig = go.Figure(data=[trace1], layout=layout);
iplot(fig);

trace1 = go.Histogram(x=feature_df[feature_df['column_var'] > 0.01]['column_var'], op
layout = dict(height=400, title='Distribution of Variable Variance > 0.01', legend=di
fig = go.Figure(data=[trace1], layout=layout);
iplot(fig);
```





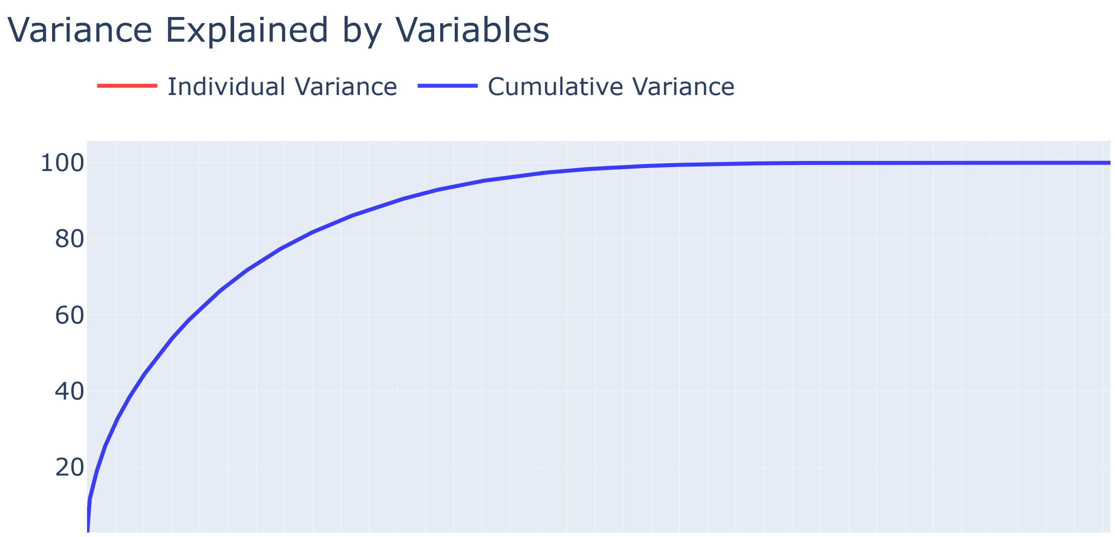
In [ ]:

```
trace1 = go.Histogram(x=feature_df['target_corr'], opacity=0.45, marker=dict(color="g
layout = dict(height=400, title='Distribution of correlation with target', legend=dic
fig = go.Figure(data=[trace1], layout=layout);
iplot(fig);
```



In [ ]:

```
# Calculating Eigenvectors and eigenvalues of Cov matirx
mean_vec = np.mean(standardized_train, axis=0)
cov_matrix = np.cov(standardized_train.T)
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)

# Create a list of (eigenvalue, eigenvector) tuples
eig_pairs = [ (np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the eigenvalue, eigenvector pair from high to low
eig_pairs.sort(key = lambda x: x[0], reverse= True)

# Calculation of Explained Variance from the eigenvalues
tot = sum(eig_vals)

# Individual explained variance
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
var_exp_real = [v.real for v in var_exp]

# Cumulative explained variance
cum_var_exp = np.cumsum(var_exp)
cum_exp_real = [v.real for v in cum_var_exp]

## plot the variance and cumulative variance
trace1 = go.Scatter(x=train.columns, y=var_exp_real, name="Individual Variance", opac
trace2 = go.Scatter(x=train.columns, y=cum_exp_real, name="Cumulative Variance", opac
layout = dict(height=400, title='Variance Explained by Variables', legend=dict(orient
fig = go.Figure(data=[trace1, trace2], layout=layout);
iplot(fig);
```


Variance Explained by Variables
Individual Variance
Cumulative Variance