Reading and Importing Data

In [1]:
```
#importing csv file
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import scale
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import classification_report, confusion_matrix
data = pd.read_csv("digit_svm.csv")
```

Data understanding and exploration

In [2]:
```
data.head()
```

Out[2]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	1	0	0	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	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	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	0	0

5 rows × 785 columns

In [3]:
```
data.shape
```

Out[3]:
```
(42000, 785)
```

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42000 entries, 0 to 41999
Columns: 785 entries, label to pixel783
dtypes: int64(785)
memory usage: 251.5 MB
```

In [5]:
```
data.describe()
```

Out[5]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
count	42000.000000	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	42000.0	...	42000.000000	42000.000000	42000.000000	42000.000000	42000.000000	42000.000000	42000.000000	42000.000000	42000.000000	42000.00
mean	4.456643	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.219286	0.117095	0.059024	0.02019	0.017228	0.001
std	2.887730	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	6.312890	4.633819	3.274488	1.75987	1.894488	0.41
min	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	2.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	4.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	7.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
max	9.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	254.000000	254.000000	253.000000	253.000000	254.000000	62.00

8 rows × 785 columns

In [6]:
```
data.isnull().sum()
```

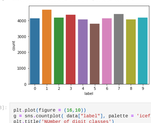
Out[6]:
```
label    0
pixel0   0
pixel1   0
pixel2   0
pixel3   0
        ..
pixel779 0
pixel780 0
pixel781 0
pixel782 0
pixel783 0
Length: 785, dtype: int64
```

Visualisation

In [7]:
```
sns.countplot(data['label'])
```

```
C:\Users\KIIT\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinte
rpretation.
  warnings.warn(
```

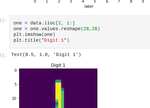
Out[7]:
```
<AxesSubplot:xlabel='label', ylabel='count'>
```



In [8]:
```
plt.plot(figure = (16,10))
g = sns.countplot( data['label'], palette = 'icefire')
plt.title('Number of digit classes')
data.label.astype('category').value_counts()
```

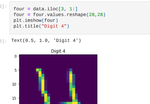
```
C:\Users\KIIT\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinte
rpretation.
  warnings.warn(
```

Out[8]:
```
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
4    4132
8    4063
0    4063
5    3795
Name: label, dtype: int64
```



In [9]:
```
one = data.iloc[2, 1:]
one = one.values.reshape(28,28)
plt.imshow(one)
plt.title('Digit 1')
```

Out[9]:
```
Text(0.5, 1.0, 'Digit 1')
```

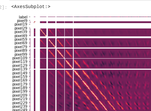


In [10]:
```
four = data.iloc[3, 1:]
four = four.values.reshape(28,28)
plt.imshow(four)
plt.title('Digit 4')
```

Out[10]:
```
Text(0.5, 1.0, 'Digit 4')
```

In [11]:
```
nine = data.iloc[11, 1:]
nine = nine.values.reshape(28,28)
plt.imshow(nine)
plt.title('Digit 9')
```

Out[11]:
```
Text(0.5, 1.0, 'Digit 9')
```



Heatmap

In [12]:
```
plt.figure(figsize=(15,15))
sns.heatmap(data=data.corr(),annot=False)
```

Out[12]:
```
<AxesSubplot:>
```

In [13]:
```
data.corr()
```

Out[13]:

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
label	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	0.033424	0.020050	0.019558	0.014490	0.009790	0.006075	NaN	NaN	NaN	NaN
pixel0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel779	0.006075	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	-0.000240	-0.000174	-0.000124	0.236633	0.905835	1.000000	NaN	NaN	NaN	NaN
pixel780	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel781	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel782	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
pixel783	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

785 rows × 785 columns

Splitting and scalling

In [14]:
```
# splitting into X and y
X = data.drop('label', axis = 1)
y = data['label']
X_scaled = scale(X)
from sklearn.model_selection import train_test_split
# train test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, train_size=0.2,test_size = 0.8, random_state = 101)
```

In [15]:
```
print('X_train shape:',X_train.shape)
print('y_train shape:',y_train.shape)
print('X_test shape:',X_test.shape)
print('y_test shape:',y_test.shape)

X_train shape: (8400, 784)
y_train shape: (8400,)
X_test shape: (33600, 784)
y_test shape: (33600,)
```

Model Building

In [16]:
```
from sklearn import svm
from sklearn import metrics
from sklearn.svm import SVC


model_linear = SVC(kernel='linear')
model_linear.fit(X_train, y_train)
y_pred = model_linear.predict(X_test)

# accuracy
print("Accuracy:", metrics.accuracy_score(y_true=y_test, y_pred=y_pred), "\n")

# cm
print(metrics.confusion_matrix(y_true=y_test, y_pred=y_pred))

accuracy: 0.913125

[[3188   0  10   5  11  20  32   3  15   1]
 [   0 3677  14  11   5   7   4   8  30   2]
 [  18  12 3175  56  19  10  19  18  37   5]
 [  13  12 104 3051   9 181   5  21  54  25]
 [   8  14  33   2 3057   4  25  31   6 110]
 [  30  23  25 136  44 2922  46  12  72  27]
 [  19  11  19   2  18  39 3113   0  10   1]
 [  24  36  49  19  36   6   0 3210  14 134]
 [  13  46  50 120  21 110  30  18 2843  21]
 [  19  17  21  22 172  29   4 161  26 2893]]
```

In [17]:
```
scores=metrics.classification_report(y_test, y_pred, labels=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print(scores)
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      3285
           1       0.95      0.98      0.96      3749
           2       0.89      0.94      0.91      3380
           3       0.88      0.88      0.88      3475
           4       0.88      0.93      0.91      3290
           5       0.87      0.86      0.87      3039
           6       0.94      0.93      0.93      3346
           7       0.92      0.92      0.92      3504
           8       0.91      0.87      0.89      3272
           9       0.90      0.86      0.88      3355

    accuracy                           0.91     33600
   macro avg       0.91      0.91      0.91     33600
weighted avg       0.91      0.91      0.91     33600
```

Non-Linear SVM

In [18]:
```
# rbf kernel with other hyperparameters kept to default
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(X_train, y_train)
```

Out[18]:
```
SVC()
```

In [19]:
```
# predict
predictions = svm_rbf.predict(X_test)

# accuracy
print(metrics.accuracy_score(y_true=y_test, y_pred=predictions))
```

```
0.9396130952380952
```