```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate
from scipy.stats import multivariate_normal as mvn


# Model classes

# Naive Bayes - class
class GaussNB():
  def fit(self, X, y, epsilon=1e-3):

    self.likelihoods = dict ()
    self.priors = dict()

    self.K = set(y.astype(int))

    for k in self.K:

      X_k = X[y == k, : ]

      self.likelihoods[k] = {'mean': X_k.mean(axis = 0), "cov": X_k.var(axis = 0) + epsilon}
      self.priors[k] = len(X-k)/len(X)


  def predict(self, X):

    N, D = X.shape

    P_hat = np.zeros((N, len(self.K)))

    for k, l in self.likelihoods.items():
      # Bayes Theorem application
      P_hat[:, k] = mvn.logpdf(X, l['mean'], l['cov']) + np.log(self.priors[k])

    return P_hat.argmax(axis = 1)

# KNN class
class KNNClassifier():

  def fit(self, X, y):
    self.X = X
    self.y = y.astype(int)

  def predict(self, X, k, epsilon = 1e-3):

    N = len(X)
    y_hat = np.zeros(N)
    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2, axis=1)
      idxt = np.argsort(dist2)[:k]
      gamma_k = 1/(np.sqrt(dist2[idxt] + epsilon))
      y_hat[i] = np.bincount(self.y[idxt], weights = gamma_k).argmax()

    return y_hat

# Gauss-Bayes class
class GaussBayes():

  def fit(self,X,y,epsilon=1e-3):

    self.likelihoods=dict()
    self.priors=dict()
    self.k = set(y.astype(int))

    for k in self.k:
      X_k = X[y==k,:]
      N_k, D = X_k.shape
      mu_k = X_k.mean(axis=0)
      self.likelihoods[k] = {"mean":X.mean(axis=0),"cov":(1/(N_k-1))*np.matmul((X_k-mu_k).T,X
      self.priors[k] = len(X_k) / len(X)

  def predict(self,X):
    N,D = X.shape
    P_hat = np.zeros((N,len(self.k)))
    for k, l in self.likelihoods.items():
      P_hat[:,k] = mvn.logpdf(X,l["mean"],l["cov"])+np.log(self.priors[k])

    return P_hat.argmax(axis=1)

# Accuracy method
def accuracy(y, y_hat):
  return np.mean(y==y_hat)

# min-max normalization method
def min_max_scaling(X):
    new_X =[]
    for i in X:
      new_X.append((i - i.min())/(i.max() - i.min()))
    return np.asarray(new_X)


# Functions for getting and viewing instances
def view_digit(x):
    """ Displays the given instance
    NumpyArray: X -> A row vector with 784 values
    """
    plt.imshow(x.reshape(28,28))
```

```
def get_mean(X, y, k):
    """ returns a row vector with the mean values for the given class
    NumpyArray: X -> An n by m matrix of instances
    NumpyArray: y -> A len(X) row vector of class labels
    Int: K        -> The label in question
    """
    return sum(X[y==k,:]/len(X[y==k,:]))

def view_mean(X, y, k):
    """ Displays the a mean representation of the given class in X
    NumpyArray: X -> An n by m matrix of instances
    NumpyArray: y -> A len(X) row vector of class labels
    Int: K        -> The label in question
    """
    view_digit(get_mean(X, y, k))
```

Data Exploration

Training data

```
# Convertion of the csv files into data frames.
path = '/content/drive/MyDrive/Enhance IT Data Science Course/Week 2/Assignment 2/Data/Mike S
data_train = pd.read_csv(path)
path = '/content/drive/MyDrive/Enhance IT Data Science Course/Week 2/Assignment 2/Data/Mike S
data_test = pd.read_csv(path)
```

```
data_train
```

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	...	774	775	776	777	778	7
0	0	0	5	0	0	0	0	0	0	0		0	0	0	0	0	
2	2	2	4	0	0	0	0	0	0	0	...	0	0	0	0	0	

Testing data

```
data_test
```

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	...	774	775	776	777	778	77
0	0	0	7	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	1	1	2	0	0	0	0	0	0	0	...	0	0	0	0	0	
2	2	2	1	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	3	3	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	4	4	4	0	0	0	0	0	0	0	...	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
9995	9995	9995	2	0	0	0	0	0	0	0	...	0	0	0	0	0	
9996	9996	9996	3	0	0	0	0	0	0	0	...	0	0	0	0	0	
9997	9997	9997	4	0	0	0	0	0	0	0	...	0	0	0	0	0	
9998	9998	9998	5	0	0	0	0	0	0	0	...	0	0	0	0	0	
9999	9999	9999	6	0	0	0	0	0	0	0	...	0	0	0	0	0	

```
X_train = data_train.to_numpy()
```

```
y_train = X_train[:, 2]
```

```
X_train = X_train[:, 3:]
```

```
X_train.shape
```

```
(60000, 784)
```

```
y_train.shape
```

```
(60000,)
```

```
X_test = data_test.to_numpy()
```

```
y_test = X_test[:, 2]
```

```
X_test = X_test[:, 3:]
```

```
X_test.shape
```

```
(10000, 784)
```

```
y_test.shape
```

```
(10000,)
```

Naive Bayes

```
model_nb = GaussNB()
```

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

```
y_pred_nb_train = model_nb.predict(X_train)
```

```
y_pred_nb_test = model_nb.predict(X_test)
```

```
acc_nb_train = accuracy(y_train, y_pred_nb_train)
acc_nb_train
```

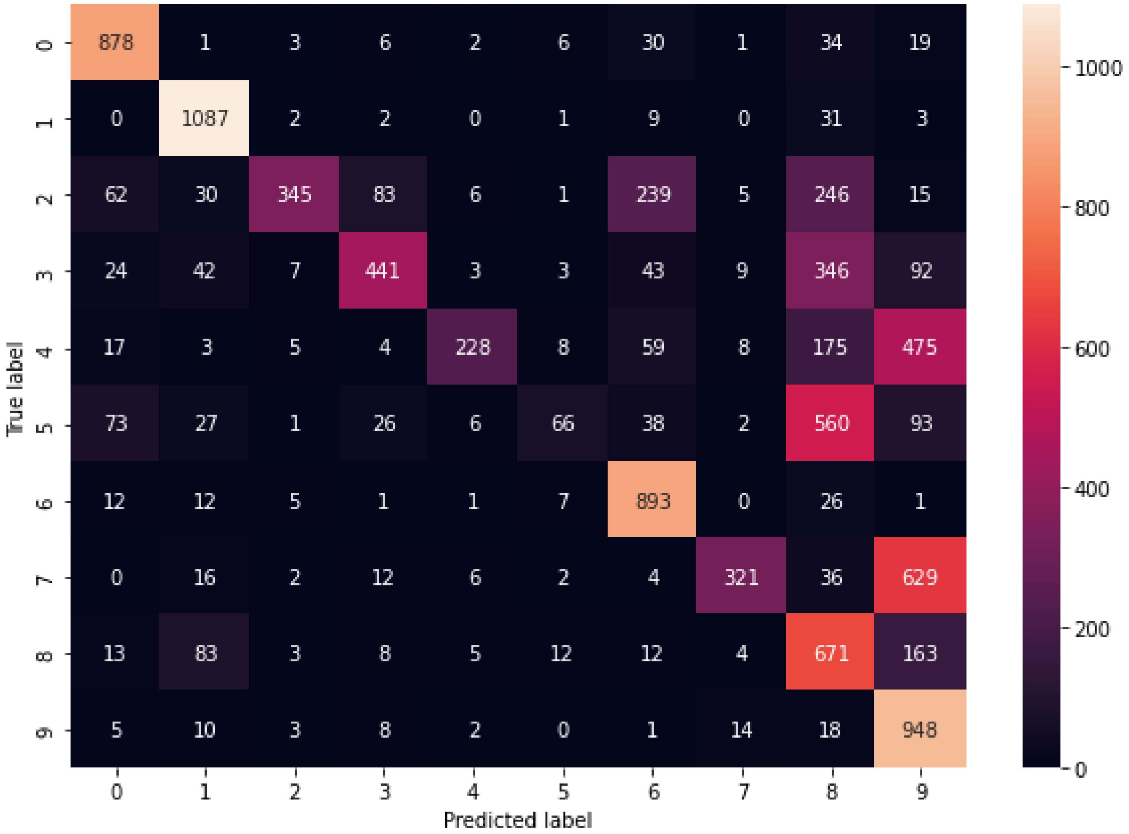
```
0.5937666666666667
```

```
acc_nb_test = accuracy(y_test, y_pred_nb_test)
acc_nb_test
```

```
0.5878
```

```
# Confusion matrix
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_nb_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```



	0	1	2	3	4	5	6	7	8	9
0	878	1	3	6	2	6	30	1	34	19
1	0	1087	2	2	0	1	9	0	31	3
2	62	30	345	83	6	1	239	5	246	15
3	24	42	7	441	3	3	43	9	346	92
4	17	3	5	4	228	8	59	8	175	475
5	73	27	1	26	6	66	38	2	560	93
6	12	12	5	1	1	7	893	0	26	1
7	0	16	2	12	6	2	4	321	36	629
8	13	83	3	8	5	12	12	4	671	163
9	5	10	3	8	2	0	1	14	18	948

Gauss-Bayes

```
model_gb = GaussBayes()
```

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

```
y_pred_gb_train = model_gb.predict(X_train)
```

```
y_pred_gb_test = model_gb.predict(X_test)
```

```
acc_gb_train = accuracy(y_train, y_pred_gb_train)
acc_gb_train
```

```
0.11651666666666667
```

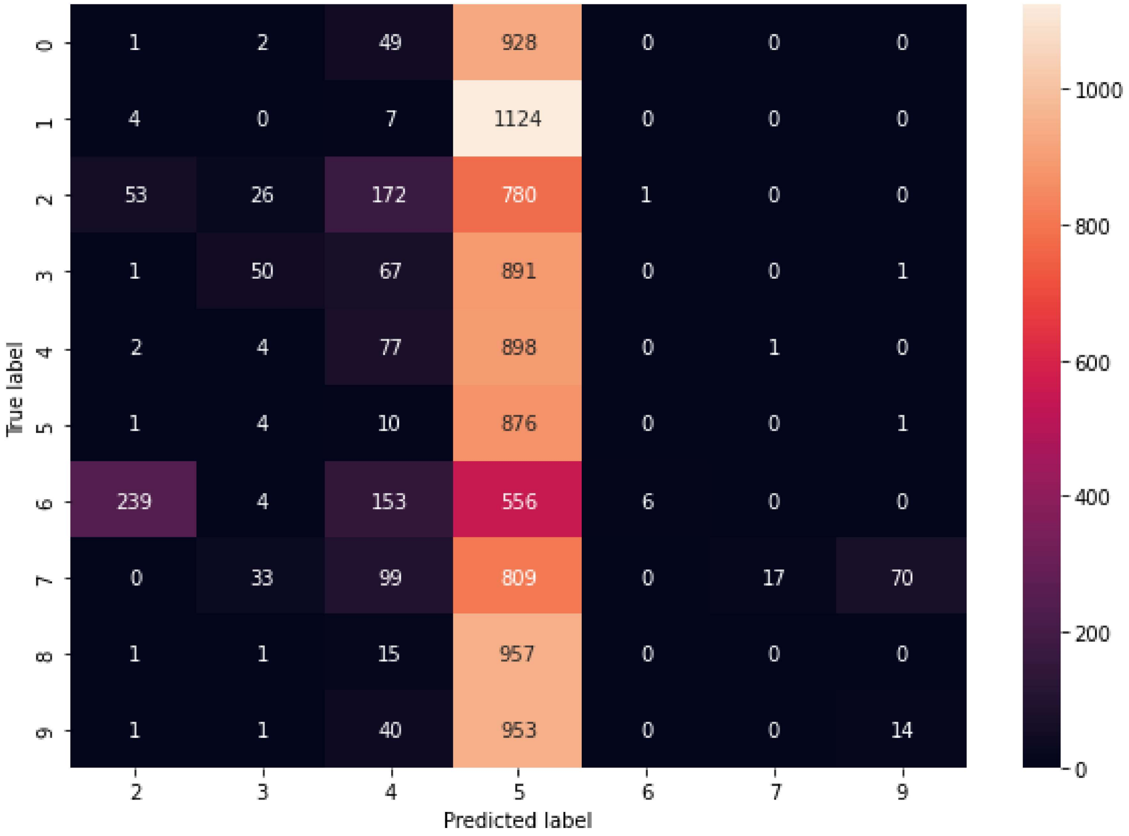
```
acc_gb_test = accuracy(y_test, y_pred_gb_test)
acc_gb_test
```

```
0.1093
```

```
# Confusion matrix
```

```
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_gb_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```



	2	3	4	5	6	7	9
0	1	2	49	928	0	0	0
1	4	0	7	1124	0	0	0
2	53	26	172	780	1	0	0
3	1	50	67	891	0	0	1
4	2	4	77	898	0	1	0
5	1	4	10	876	0	0	1
6	239	4	153	556	6	0	0
7	0	33	99	809	0	17	70
8	1	1	15	957	0	0	0
9	1	1	40	953	0	0	14

KNN

```
model_knn = KNNClassifier()
```

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

```
y_pred_knn_train = model_knn.predict(X_train, 12)
```

```
y_pred_knn_test = model_knn.predict(X_test, 12)
```

```
acc_knn_train = 'Soon!'
acc_knn_test = 'Soon!'
```

```
acc_knn_train = accuracy(y_train, y_pred_knn_train)
acc_knn_train
```

```
1.0
```

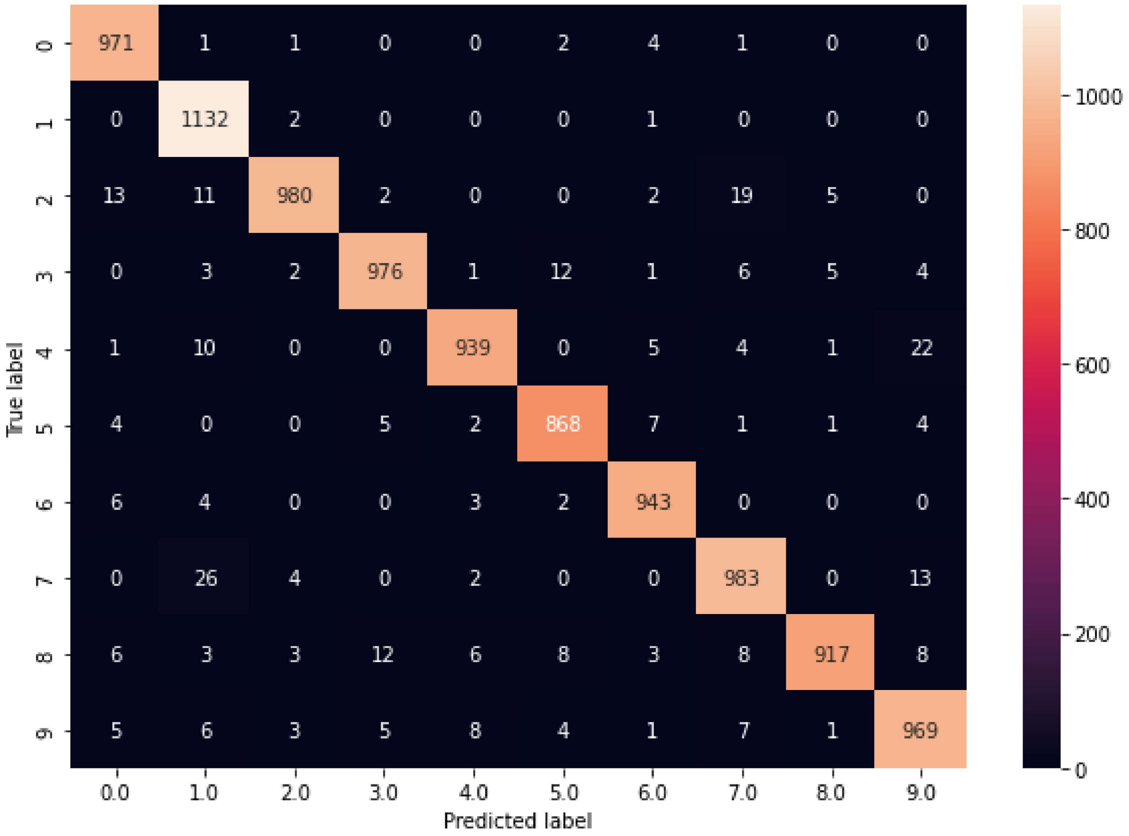
```
acc_knn_test = accuracy(y_test, y_pred_knn_test)
acc_knn_test
```

```
0.9678
```

```
# Confusion matrix

plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_knn_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```



True label	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
0	971	1	1	0	0	2	4	1	0	0
1	0	1132	2	0	0	0	1	0	0	0
2	13	11	980	2	0	0	2	19	5	0
3	0	3	2	976	1	12	1	6	5	4
4	1	10	0	0	939	0	5	4	1	22
5	4	0	0	5	2	868	7	1	1	4
6	6	4	0	0	3	2	943	0	0	0
7	0	26	4	0	2	0	0	983	0	13
8	6	3	3	12	6	8	3	8	917	8
9	5	6	3	5	8	4	1	7	1	969

Model comparison

```
table = [['Model', 'Train', 'Test'],
         ['Naive Bayes', acc_nb_train, acc_nb_test],
         ['Gauss-Bayes', acc_gb_train, acc_gb_test],
         ['KNN', acc_knn_train, acc_knn_test]]
```

```
print(tabulate(table))
```

```
-----------  -------------------  ------
Model        Train                Test
Naive Bayes  0.5937666666666667   0.5878
Gauss-Bayes  0.11651666666666667  0.1093
KNN          1.0                  0.9678
-----------  -------------------  ------
```

Data transormation - normalization

```
norm_X_train = min_max_scaling(X_train)
norm_X_test = min_max_scaling(X_test)
```

Naive Bayes

```
model_nb = GaussNB()
```

```
model_nb.fit(norm_X_train, y_train)
```

```
y_pred_nb_train = model_nb.predict(norm_X_train)
```

```
y_pred_nb_test = model_nb.predict(norm_X_test)
```

```
acc_nb_train= accuracy(y_train, y_pred_nb_train)
acc_nb_train
```

```
0.7682333333333333
```

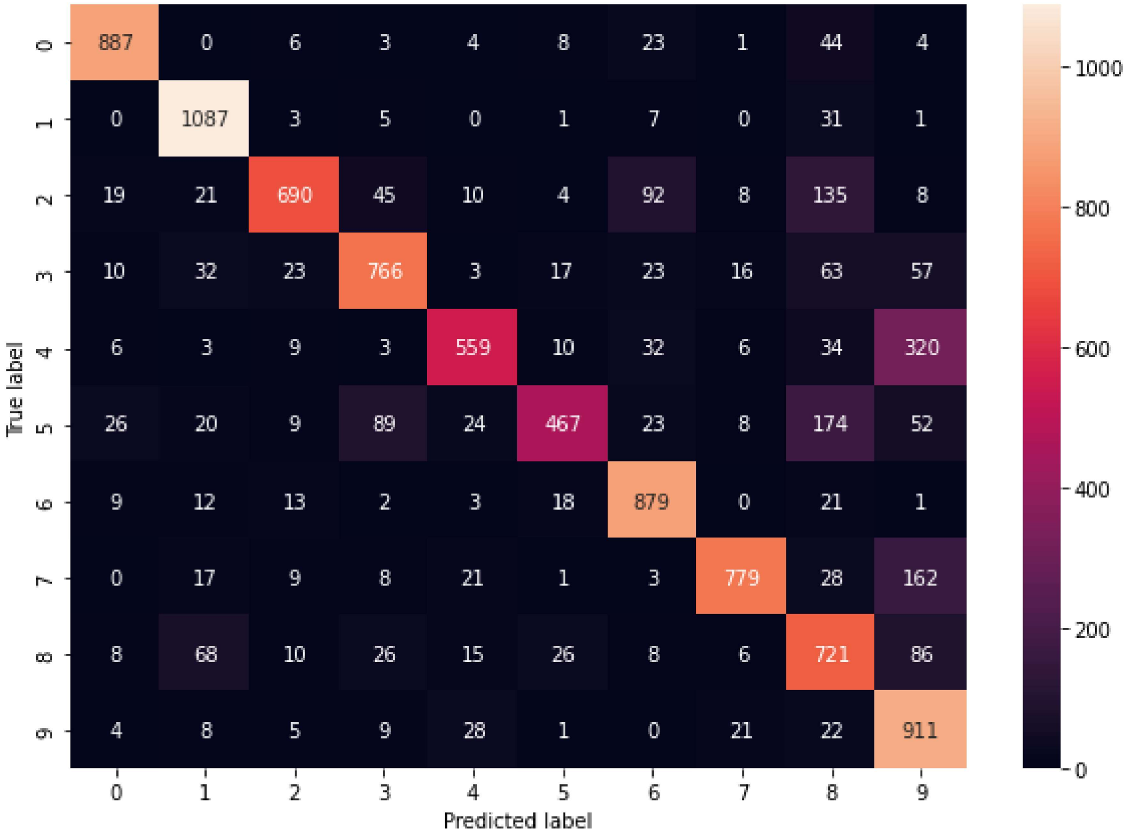
```
acc_nb_test= accuracy(y_test, y_pred_nb_test)
acc_nb_test
```

```
0.7746
```

```
# Confusion matrix
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_nb_test, name='Predicted')
```

```
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```



	0	1	2	3	4	5	6	7	8	9
0	887	0	6	3	4	8	23	1	44	4
1	0	1087	3	5	0	1	7	0	31	1
2	19	21	690	45	10	4	92	8	135	8
3	10	32	23	766	3	17	23	16	63	57
4	6	3	9	3	559	10	32	6	34	320
5	26	20	9	89	24	467	23	8	174	52
6	9	12	13	2	3	18	879	0	21	1
7	0	17	9	8	21	1	3	779	28	162
8	8	68	10	26	15	26	8	6	721	86
9	4	8	5	9	28	1	0	21	22	911

Gauss-Bayes

```
model_gb = GaussBayes()
```

```
model_gb.fit(norm_X_train, y_train)
```

```
y_pred_gb_train = model_gb.predict(norm_X_train)
```

```
y_pred_gb_test = model_gb.predict(norm_X_test)
```

```
acc_gb_train = accuracy(y_train, y_pred_gb_train)
acc_gb_train
```

```
0.8657333333333334
```

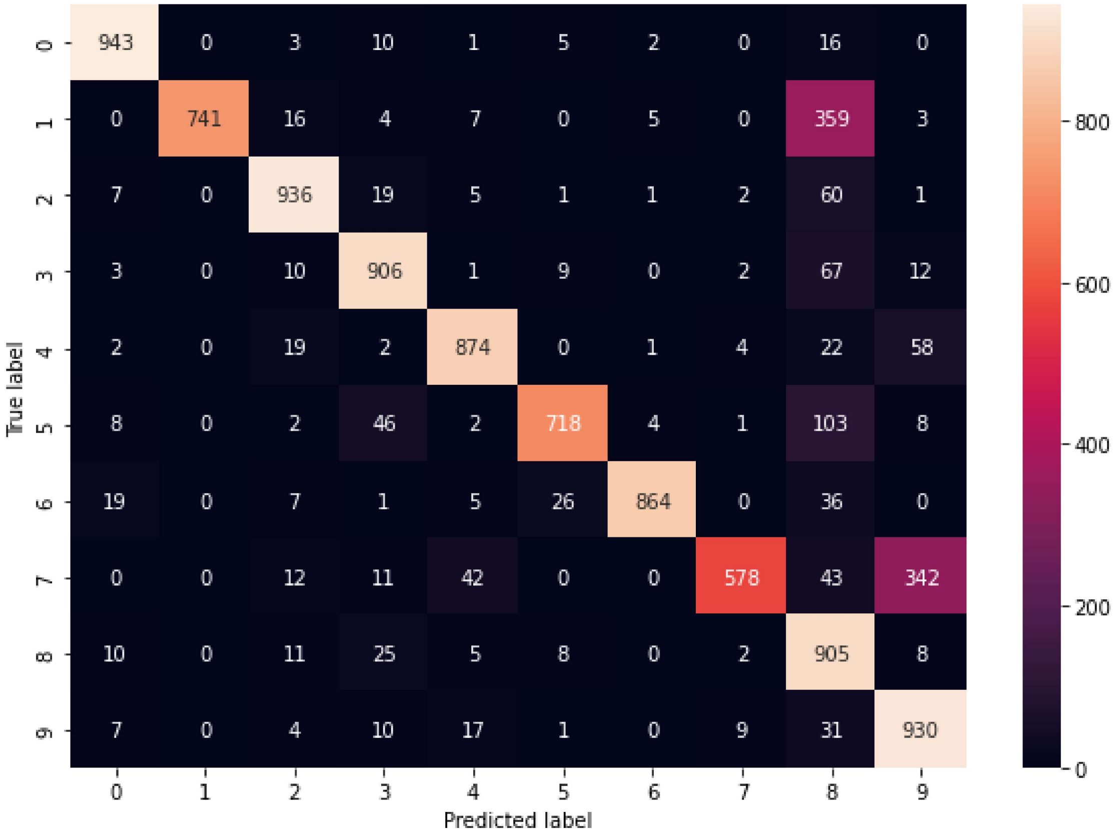
```
acc_gb_test = accuracy(y_test, y_pred_gb_test)
```

```
acc_gb_test
```

```
    0.8395
```

```
# Confusion matrix
plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_gb_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
    Text(0.5, 42.0, 'Predicted label')
```



KNN

```
model_knn = KNNClassifier()
```

```
model_knn.fit(norm_X_train, y_train)
```

```
y_pred_knn_train = model_knn.predict(norm_X_train, 12)
```

```
y_pred_knn_test = model_knn.predict(norm_X_test, 12)
```

```
acc_knn_train = 'Soon!'
acc_knn_test = 'Soon!'
```

```
acc_knn_train = accuracy(y_train, y_pred_knn_train)
acc_knn_train
```

```
1.0
```

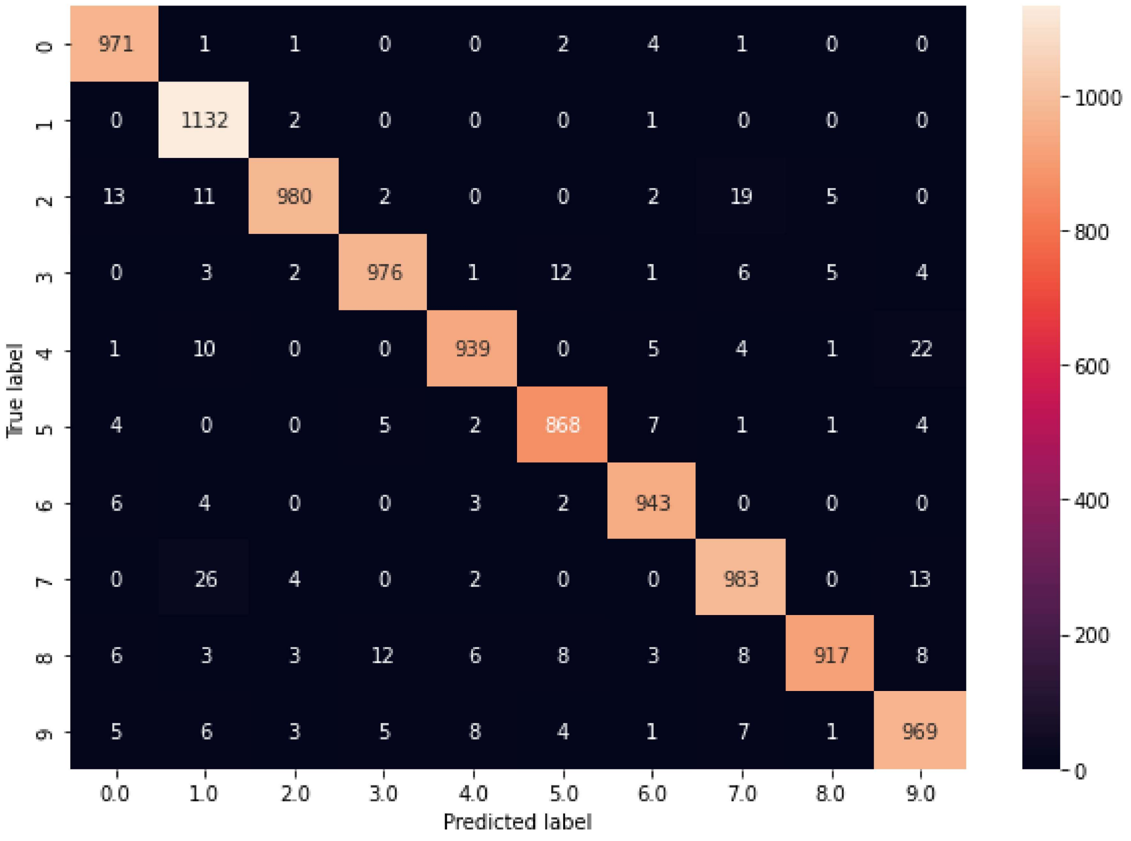
```
acc_knn_test = accuracy(y_test, y_pred_knn_test)
acc_knn_test
```

```
0.9678
```

```
# Confusion matrix

plt.figure(figsize=(10,7))
y_actu = pd.Series(y_test, name='Actual')
y_pred = pd.Series(y_pred_knn_test, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Text(0.5, 42.0, 'Predicted label')
```



	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
0	971	1	1	0	0	2	4	1	0	0
1	0	1132	2	0	0	0	1	0	0	0
2	13	11	980	2	0	0	2	19	5	0
3	0	3	2	976	1	12	1	6	5	4
4	1	10	0	0	939	0	5	4	1	22
5	4	0	0	5	2	868	7	1	1	4
6	6	4	0	0	3	2	943	0	0	0
7	0	26	4	0	2	0	0	983	0	13
8	6	3	3	12	6	8	3	8	917	8
9	5	6	3	5	8	4	1	7	1	969

Model comparison

```
table = [['Model', 'Train', 'Test'],
         ['Naive Bayes', acc_nb_train, acc_nb_test],
         ['Gauss-Bayes', acc_gb_train, acc_gb_test],
         ['KNN', acc_knn_train, acc_knn_test]]


print(tabulate(table))
```

```
-----------  ------------------  ------
Model        Train               Test
Naive Bayes  0.7682333333333333  0.7746
Gauss-Bayes  0.8657333333333334  0.8395
KNN          1.0                 0.9678
-----------  ------------------  ------
```

✓ 0s completed at 7:03 PM