



MTL782 DATA MINING

Assignment 2

Team Members

Khushvind Maurya	2021MT10238
Aniket Singh	2021MT10256
Rishabh Kumar Jaiswal	2021MT10924

Contents

1	Importing And Preprocessing Data	3
1.1	Libraries Used	3
1.2	Importing The Libraries	3
1.3	Downloading The Dataset	3
1.4	Normalising The Dataset	3
1.5	Flattening the Data so that it can be used for Training	3
2	Decision Tree	4
2.1	Implementing a Decision Tree Classifier, and checking the accuracy using k-fold Cross Validation.	4
2.2	Evaluating the method using different evaluation metrics.	4
2.3	Parameter Tuning through Grid Search	4
2.4	Parameter Tuning through Random Search	5
3	Random Forest	6
3.1	Implementing a Random Forest Classifier, and checking the accuracy using k-fold Cross Validation.	6
3.2	Evaluating the method using different evaluation metrics.	6
3.3	Parameter Tuning through Grid Search	6
3.4	Parameter Tuning through Random Search	7
4	Naive Bayes	8
4.1	Implementing a Naive Bayes Classifier, and checking the accuracy using k-fold Cross Validation.	8
4.2	Evaluating the method using different evaluation metrics.	8
4.3	Parameter Tuning through Grid and Random Search	8
5	KNN	9
5.1	Implementing a KNN Classifier, and checking the accuracy using k-fold Cross Validation.	9
5.2	Evaluating the method using different evaluation metrics.	9
5.3	Parameter Tuning through Grid Search	9
5.4	Parameter Tuning through Random Search	9
6	Neural Network	11
6.1	Implementing a Neural Network Classifier, and checking the accuracy using k-fold Cross Validation.	11
6.2	Evaluating the method using different evaluation metrics.	11
6.3	Parameter Tuning through Grid Search	11
6.4	Parameter Tuning through Random Search	12
7	Observations	13

Link to Colab Notebook: https://colab.research.google.com/drive/1Qi0leW7dT9dhK0sri5Hn54B08N_Wu8Iq?usp=sharing

1 Importing And Preprocessing Data

1.1 Libraries Used

1. Scikit-learn (For Implementing the Models, and performing evaluation)
2. Tensorflow (For Importing The dataset)
3. Matplotlib (For Plotting the digits)

1.2 Importing The Libraries

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from sklearn.metrics import classification_report, accuracy_score, precision_score
from sklearn.model_selection import cross_val_score, GridSearchCV, RandomizedSearchCV
import matplotlib.pyplot as plt
```

1.3 Downloading The Dataset

```
[2] (X_train, y_train), (X_test, y_test) = mnist.load_data()

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
```

1.4 Normalising The Dataset

```
[4] X_train = X_train.astype('float32') / 255.0
    X_test = X_test.astype('float32') / 255.0
```

1.5 Flattening the Data so that it can be used for Training

```
X_train_flat = X_train.reshape(X_train.shape[0], -1)
X_test_flat = X_test.reshape(X_test.shape[0], -1)
```

2 Decision Tree

2.1 Implementing a Decision Tree Classifier, and checking the accuracy using k-fold Cross Validation.

```
[8] from sklearn.tree import DecisionTreeClassifier

A. Implementing a Decision Tree Classifier, and checking the accuracy using k-fold Cross Validation.

[9] model = DecisionTreeClassifier()
cross_val = (cross_val_score(model,X_train_flat,y_train,cv= 5))
print ("Cross Validation Score =",cross_val.mean())

Cross Validation Score = 0.8656666666666666
```

2.2 Evaluating the method using different evaluation metrics.

```
[10] model.fit(X_train_flat,y_train)
predictions = model.predict(X_test_flat)
print (classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.92	0.93	0.93	980
1	0.95	0.96	0.96	1135
2	0.87	0.86	0.86	1032
3	0.84	0.85	0.84	1010
4	0.87	0.88	0.87	982
5	0.82	0.84	0.83	892
6	0.89	0.89	0.89	958
7	0.92	0.91	0.91	1028
8	0.83	0.80	0.82	974
9	0.85	0.85	0.85	1009
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

2.3 Parameter Tuning through Grid Search

For parameter tuning using Grid Search, we tune for the parameters in the grid, 'max_depth': [None, 2, 4, 6, 8], 'min_samples_split': [2, 4, 6, 8]

```
[17] model = DecisionTreeClassifier()
param_grid = {'max_depth': [None, 2, 4, 6, 8],
              'min_samples_split': [2, 4, 6, 8],
              }

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_flat, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)

Best Parameters: {'max_depth': None, 'min_samples_split': 6}
Best Score: 0.8665666666666667
```

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'max_depth': None, 'min_samples_split': 6

2.4 Parameter Tuning through Random Search

For parameter tuning using Random Search, we tune for the parameters in the grid, 'max_depth': [None, 2, 4, 6, 8], 'min_samples_split': [2, 4, 6, 8]

```
[18] randomized_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=10, cv=5, scoring='accuracy')
    randomized_search.fit(X_train_flat, y_train)

    best_params = randomized_search.best_params_
    best_score = randomized_search.best_score_

    print("Best Parameters:", best_params)
    print("Best Score:", best_score)

Best Parameters: {'min_samples_split': 2, 'max_depth': None}
Best Score: 0.8668000000000001
```

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'max_depth': None, 'min_samples_split': 2

3 Random Forest

3.1 Implementing a Random Forest Classifier, and checking the accuracy using k-fold Cross Validation.

```
from sklearn.ensemble import RandomForestClassifier
```

A. Implementing a random Forest Classifier, and checking the accuracy using k-fold Cross Validation.

```
[12] model = RandomForestClassifier()
      cross_val = (cross_val_score(model,X_train_flat,y_train,cv= 5))
      print ("Cross Validation Score =",cross_val.mean())
```

Cross Validation Score = 0.96595

3.2 Evaluating the method using different evaluation metrics.

```
[13] model.fit(X_train_flat,y_train)
      predictions = model.predict(X_test_flat)
      print (classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.96	0.97	0.96	1032
3	0.96	0.96	0.96	1010
4	0.97	0.97	0.97	982
5	0.97	0.96	0.96	892
6	0.97	0.98	0.97	958
7	0.97	0.96	0.97	1028
8	0.96	0.96	0.96	974
9	0.96	0.95	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

3.3 Parameter Tuning through Grid Search

For parameter tuning using Grid Search, we tune for the parameters in the grid, 'max_depth' : [None, 2, 4, 6, 8], 'n_estimators': [80, 90, 100, 110, 120]

```
[9] model = RandomForestClassifier()
      param_grid = {'max_depth': [None, 2, 4, 6, 8],
                    'n_estimators': [80, 90, 100, 110, 120]}

      grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
      grid_search.fit(X_train_flat, y_train)

      best_params = grid_search.best_params_
      best_score = grid_search.best_score_

      print("Best Parameters:", best_params)
      print("Best Score:", best_score)
```

Best Parameters: {'max_depth': None, 'n_estimators': 120}
Best Score: 0.96685

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'max_depth' : None, 'n_estimators': 120

3.4 Parameter Tuning through Random Search

For parameter tuning using Random Search, we tune for the parameters in the grid, 'max_depth' : [None, 2, 4, 6, 8], 'n_estimators': [80, 90, 100, 110, 120]

```
[10] randomized_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=5, cv=5, scoring='accuracy')
    randomized_search.fit(X_train_flat, y_train)

    best_params = randomized_search.best_params_
    best_score = randomized_search.best_score_

    print("Best Parameters:", best_params)
    print("Best Score:", best_score)

Best Parameters: {'n_estimators': 90, 'max_depth': None}
Best Score: 0.9663833333333333
```

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'max_depth' : None, 'n_estimators': 90

4 Naive Bayes

4.1 Implementing a Naive Bayes Classifier, and checking the accuracy using k-fold Cross Validation.

```
[14] from sklearn.naive_bayes import GaussianNB
```

A. Implementing a Naive Bayes Classifier, and checking the accuracy using k-fold Cross Validation.

```
[15] model = GaussianNB()
      cross_val = (cross_val_score(model,X_train_flat,y_train,cv= 5))
      print ("Cross Validation Score =",cross_val.mean())
```

Cross Validation Score = 0.5617666666666666

4.2 Evaluating the method using different evaluation metrics.

```
[16] model.fit(X_train_flat,y_train)
      predictions = model.predict(X_test_flat)
      print (classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.79	0.89	0.84	980
1	0.85	0.95	0.90	1135
2	0.90	0.26	0.40	1032
3	0.71	0.35	0.47	1010
4	0.88	0.17	0.29	982
5	0.55	0.05	0.09	892
6	0.65	0.93	0.77	958
7	0.88	0.27	0.42	1028
8	0.28	0.67	0.40	974
9	0.37	0.95	0.53	1009
accuracy			0.56	10000
macro avg	0.69	0.55	0.51	10000
weighted avg	0.69	0.56	0.52	10000

4.3 Parameter Tuning through Grid and Random Search

Since Naive Bayes models in scikit-learn do not typically have hyperparameters to tune through traditional methods like grid search or random search, we have not shown it here.

5 KNN

5.1 Implementing a KNN Classifier, and checking the accuracy using k-fold Cross Validation.

```
[17] from sklearn.neighbors import KNeighborsClassifier

A. Implementing a KNN Classifier, and checking the accuracy using k-fold Cross Validation.

[18] model = KNeighborsClassifier()
      cross_val = (cross_val_score(model,X_train_flat,y_train,cv= 5))
      print ("Cross Validation Score =",cross_val.mean())

Cross Validation Score = 0.9692833333333335
```

5.2 Evaluating the method using different evaluation metrics.

```
model.fit(X_train_flat,y_train)
predictions = model.predict(X_test_flat)
print (classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.98	980
1	0.95	1.00	0.98	1135
2	0.98	0.96	0.97	1032
3	0.96	0.97	0.97	1010
4	0.98	0.96	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.99	0.98	958
7	0.96	0.96	0.96	1028
8	0.99	0.94	0.96	974
9	0.96	0.95	0.95	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

5.3 Parameter Tuning through Grid Search

For parameter tuning using Grid Search, we tune for the parameters in the grid, 'n_neighbors':
[1,2,3,4,5,6,7,8,9,10]

```
model = KNeighborsClassifier()
param_grid = {'n_neighbors': [i for i in range(1,11)]}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_flat, y_train)

best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)

Best Parameters: {'n_neighbors': 3}
Best Score: 0.9700666666666666
```

Best k-fold Cross Validation accuracy is obtained for the hyperparameters 'n_neighbors':

3

5.4 Parameter Tuning through Random Search

For parameter tuning using Random Search, we tune for the parameters in the grid, 'n_neighbors':
[1,2,3,4,5,6,7,8,9,10]

```
[13] randomized_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=5, cv=5, scoring='accuracy')
    randomized_search.fit(X_train_flat, y_train)

    best_params = randomized_search.best_params_
    best_score = randomized_search.best_score_

    print("Best Parameters:", best_params)
    print("Best Score:", best_score)

Best Parameters: {'n_neighbors': 3}
Best Score: 0.9700666666666666
```

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'n_neighbors':

3

6 Neural Network

6.1 Implementing a Neural Network Classifier, and checking the accuracy using k-fold Cross Validation.

```
[20] from sklearn.neural_network import MLPClassifier
```

A. Implementing a Neural Network Classifier, and checking the accuracy using k-fold Cross Validation.

```
[21] model = MLPClassifier()  
cross_val = (cross_val_score(model,X_train_flat,y_train,cv= 5))  
print ("Cross Validation Score =",cross_val.mean())
```

```
Cross Validation Score = 0.96115
```

6.2 Evaluating the method using different evaluation metrics.


```
[22] model.fit(X_train_flat,y_train)  
predictions = model.predict(X_test_flat)  
print (classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.96	0.98	0.97	980
1	0.99	0.99	0.99	1135
2	0.96	0.95	0.96	1032
3	0.96	0.96	0.96	1010
4	0.97	0.96	0.97	982
5	0.96	0.96	0.96	892
6	0.97	0.97	0.97	958
7	0.97	0.96	0.97	1028
8	0.94	0.95	0.95	974
9	0.96	0.96	0.96	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	10000

6.3 Parameter Tuning through Grid Search

For parameter tuning using Grid Search, we tune for the parameters in the grid, 'hidden_layer_sizes': [(50,), (100,), (200,), (50,50), (10,10)]

```
model = MLPClassifier()  
param_grid = {'hidden_layer_sizes': [(50,), (100,), (200,), (50,50), (10,10)]}  
  
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')  
grid_search.fit(X_train_flat, y_train)  
  
best_params = grid_search.best_params_  
best_score = grid_search.best_score_  
  
print("Best Parameters:", best_params)  
print("Best Score:", best_score)
```

 /usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: Converge
warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: Converge
warnings.warn(
Best Parameters: {'hidden_layer_sizes': 200}
Best Score: 0.9709333333333333

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'hidden_layer_sizes': (200,)

6.4 Parameter Tuning through Random Search

For parameter tuning using Random Search, we tune for the parameters in the grid, 'hidden_layer_sizes': [(50,), (100,), (200), (50,50), (10,10)]

```
randomized_search = RandomizedSearchCV(estimator=model, param_distributions=param_grid, n_iter=3, cv=5, scoring='accuracy')
randomized_search.fit(X_train_flat, y_train)

best_params = randomized_search.best_params_
best_score = randomized_search.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

/usr/local/lib/python3.10/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached. Optimization terminated. The current parameters are: {'hidden_layer_sizes': (50, 50)}.

Best Parameters: {'hidden_layer_sizes': (50, 50)}

Best Score: 0.9543166666666666

Best k-fold Cross Validation accuracy is obtained for the hyperparameters, 'hidden_layer_sizes': (50,50)

7 Observations

A.

Model	Accuracy
Decision Tree	0.8657
Random Forest	0.96595
Naive Bayes	0.56177
KNN	0.969283
Neural Network	0.96115

Table 1: K-Fold Cross Validation

B

Model	Accuracy	F1 Score	Precision	Recall
Decision Tree	0.88	0.88	0.88	0.88
Random Forest	0.97	0.97	0.97	0.97
Naive Bayes	0.56	0.51	0.69	0.55
KNN	0.97	0.97	0.97	0.97
Neural Network	0.97	0.97	0.97	0.97

Table 2: Different Evaluation Metrics

C

Model	Grid Search	Random Search
Decision Tree	0.86657	0.8668
Random Forest	0.96685	0.966383
Naive Bayes	NA	NA
KNN	0.97007	0.97007
Neural Network	0.97093	0.954317

Table 3: Parameter Tuning through Grid Search/Random Search (Accuracy)