

Classification using ML methods and Neural Networks

by Gowri Pamarthi

Submission date: 29-Mar-2023 05:43PM (UTC+0100)

Submission ID: 200917979

File name:

42643_Gowri_Pamarthi_Classification_using_ML_methods_and_Neural_Networks_495066_1907856348.docx
(1.27M)

Word count: 4341

Character count: 23885



Student Name: GOWRI PAMARTHI

Student Number: 2206771

Module Name: Application of Machine Learning (ACCA7025)

Assignment Title: Classification using ML methods and Neural Networks

Course work 1

Table of content

Table of content	2
List of Figures	3
List of Listings	3
List of Abbreviations	4
1 Introduction	5
1.1 Background of study	5
1.2 Purpose of study	5
1.3 Aim and Objective	5
2 Literature Review	6
2.1 Recent Studies	6
2.2 Role of Machine Learning and TensorFlow techniques of MNIST Data	6
2.3 Related Work	6
3 Methodology	8
3.1 Model Building using Machine Learning	8
3.1.1 Random Forest	8
3.1.2 Decision Tree	9
3.2 Model Building using TensorFlow	10
3.3 Evaluation Matrices	10
4 Result and Discussion	11
4.1 Data Description	11
4.2 Import Packages	11
4.3 Data Loading	11
4.4 Data Preprocessing	12
4.5 Data Visualization	14
4.6 Model Creation	15
4.7 Final Result	20

5	Conclusion	21
5.1	Conclusion	21
5.2	Recommendation and Future Scope	21
6	Reference	22
7	Appendix.....	24
8	Video Link	29

List of Figures

Figure 3.1	Working process of Random Forest algorithm.	8
Figure 3.2	Working process of Decision Tree algorithm.	9
Figure 3.3	Working process of TensorFlow	10
Figure 4.1	Data information about the First dataset.	12
Figure 4.2	Data information in the Second dataset.	12
Figure 4.3	Check how many nan values are present in both datasets.	13
Figure 4.4	Find the minimum and maximum unique values in this dataset.	13
Figure 4.5	Find how many duplicate values are present in this dataset.	13
Figure 4.6	Check whether any Null values are present or not.	13
Figure 4.7	Display the Total number of unique values of this outcome column.	13
Figure 4.8	Shows all the column names of this project.	13
Figure 4.9	Represent the information about the target variable graphically.	14
Figure 4.10	Display All the input images.	15
Figure 4.11	Display the shape of the train and test data.	16
Figure 4.12	Called random forest classifier.	16
Figure 4.13	Accuracy of the RF model.	16
Figure 4.14	Called Decision Tree Classifier.	17
Figure 4.15	Accuracy of DT	17
Figure 4.16		18
Figure 4.16		19
Figure 4.16		20

List of Listings

Listing 4.1	Provide the information about required Libraries.	11
-------------	---	----

Listing 4.2 Process of Data Loading	11
Listing 4.3 shows all the required libraries for running the model.	15
Listing 4.4 Dividing the dataset.	15
Listing 4.5	18
Listing 4.5	18
Listing 4.5	18
Listing 4.5	19

List of Abbreviations

Name	Description
ML	Machine Learning
DL	Deep Learning
RF	Random Forest
DT	Decision Tree
SVM	Support Vector Machine
TF	TensorFlow

1 Introduction

1.1 Background of study

The fashion-MNSIT dataset uses this data to recognize clothing items in images as online shopping grows. Artificial intelligence methods that can recognize and categorize human clothing are required and can be applied to increase sales or comprehend users better. It can be used to compare machine learning techniques because it uses the same image size, data format, and training/testing split structure as the original MNIST database. using TensorFlow to identify the clothing after using machine learning techniques to analyze this dataset.

1.2 Purpose of study

The Fashion-MNIST challenge is one of the classification issues that has been addressed by machine learning. Here, the author uses machine learning methods to attempt to identify the clothing. The fashion industry is currently expanding daily, bringing new brands to market. Classifying the items is challenging, but it is possible to identify clothing using machine learning algorithms. These factors led to the Fashion MNIST dataset being utilized to train machine learning algorithms to create the ideal model for image classification.

1.3 Aim and Objective

Aim: The main aim of this project is to classify the Fashion MNIST data using Machine learning and TensorFlow.

Objective:

- Analyzed the Fashion data for getting more information with the help of statistical and graphical ways.
- Predict the model with the help of Machine Learning and the TensorFlow technique.
- Evaluate performance and based on the performance select one model and finalize the model.

2 Literature Review

2.1 Recent Studies

This fashion MNIST data is one of the most popular datasets for the classification of different fashion items using multiple techniques. In this case, the author uses ML techniques, but many authors successfully compile it with other techniques also such as DL CNN techniques. Vijayaraj *et al* use convolution neural network (CNN) techniques to get a more accurate result and they got almost 90 percent accuracy from it [1]. But Leithardt (2021) uses the same CNN techniques and got an accuracy of 89 percent [2] and then uses ML SVM techniques which he got an accuracy of 99.1 percent, which is a very well-trained model for this project.

2.2 Role of Machine Learning and TensorFlow techniques of MNIST Data

Machine learning played a vital role in every data for predicting the model. Similarly, TensorFlow also does the same thing but compared to ML it gives more accurate results. Greeshma and Sreekumar used SVM to classify the MNIST data [3]. According to their study, they got 86.53 %. But in this project, the author did not use these techniques because it takes more time, and it gives less accuracy. According to the dataset, it comes under multiclass classification so all the ensemble models can be capable to handle this technique. Compared to SVM, RF gives more accuracy.

Similarly, TensorFlow also provides good accuracy because it has come under deep learning techniques [4]. Compare to ML it teaches models in a better way for this reason accuracy always increases.

2.3 Related Work

Fashion MNIST is one of the popular datasets of image processing. Many programmers used this dataset to process the image. ML and DL both techniques are used to find the best prediction. Using different types of scaling of linear models as well as deep learning in the UK biobank to process the brain data [5]. Initially, they used MNIST Fashion to predict the accuracy using the deep-nonlinear model. After that test this model of their brain image data. Kadam doing the same project, but they used CNN techniques to classify images [6]. After completing their execution, they got 99% Accuracy. Therefore, it means that different types of algorithms give different outcomes.

At the same time, Kavadi doing the same project but they use garment data to classify Fashion MNIST, but they used LeNet-5 Architecture from the CNN model [7]. After the experiment, they got 98% accuracy, and using this model they try to classify the clothes. However, Leithardt

also uses CNN techniques to classify the image data [8]. The main motive of this project is to increase the clothing sale of the garment company. Leithardt used the SVM model, but it provides only 89.7% at the same time Leithardt got 99.1% after using CNN dropout-3 techniques. therefore, after studying both techniques the author understands that different CNN model provides different accuracy percentages.

3 Methodology

In this chapter, the author describes all the algorithms which are already used in this project. ML and DL are both techniques used here for finding the best predictive result. ML, RF, and DT are used and DL TensorFlow methods are used. Below the author tries to provide valuable information about all the mentioned techniques.

3.1 Model Building using Machine Learning

Based on the target variable this problem comes under multiclass classification. For this reason, the author used RF and DT for solving the multiclass problem. Because both algorithms are capable of handling multiclassification properly.

3.1.1 Random Forest

Random forest (RF) is one of the popular algorithms compared to the other algorithms and most of the time it gives high accuracy [9]. For this reason, many programmers or researchers used this algorithm. It is also capable of handling classification as well as regression of both problems. MNIST is a classification problem so the author used a classifier here. The entire working procedure is explained in Figure 3.1.

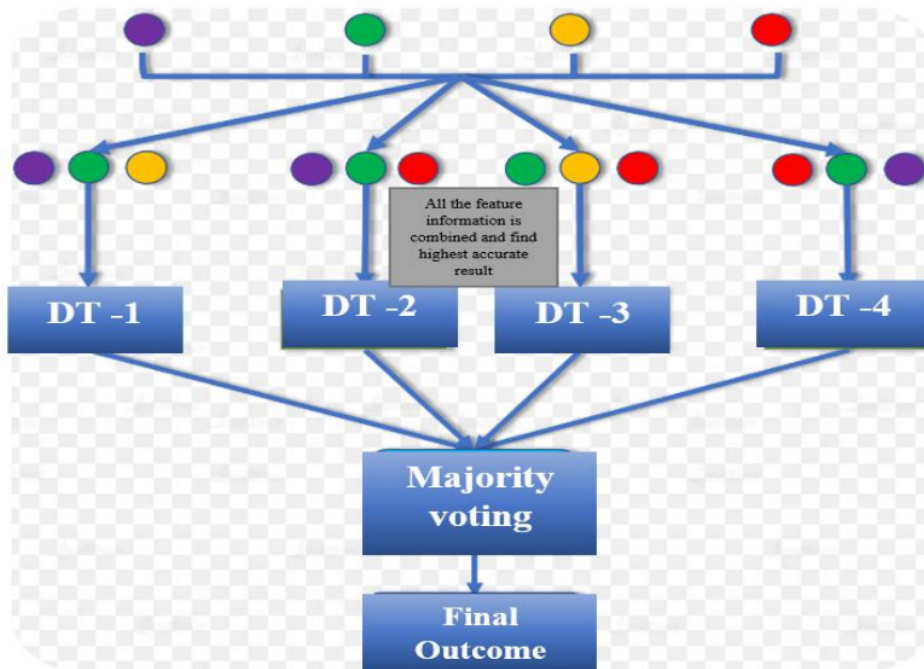


Figure 3.1 Working process of Random Forest algorithm.

Everyone knows that RF is a combination of DT. It creates many DTs to execute the program. After running, the entire DT model takes majority voting and based on the majority voting it takes decision means it gives the final result. It always tries to reduce overfitting and increase the accuracy of the model. If missing values are present in the data, then also it is capable to gives a more accurate model. But it takes more time because inside the model many DTs are running at a time.

3.1.2 Decision Tree

Decision trees are one of the most popular machine-learning techniques due to their simplicity and ease to understand. In decision analysis, a decision tree can be used to represent decisions and decision-making. The author also gives the information about working process of DT in Figure 3.2.

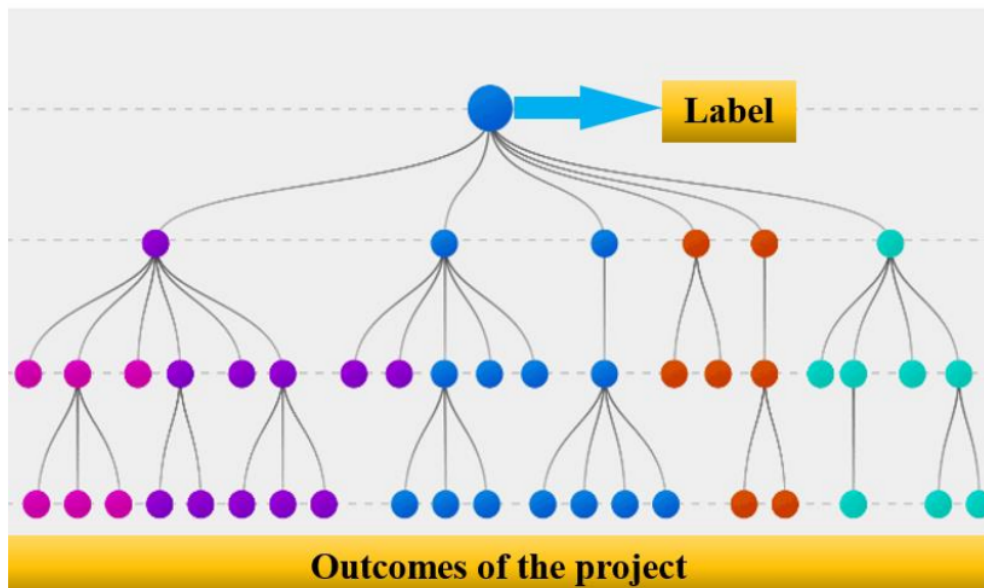


Figure 3.2 Working process of Decision Tree algorithm.

Based on the target column the DT starts the interpretation and after that, it used the features of this data to provide the outcomes. The advantages of DT are, it is interpretable and nonparametric [10]. The decision tree method requires less effort for data preparation during pre-processing. Usually, this results in the data being overfitting, where it produces inaccurate predictions. A small change in the data can have a significant impact on the decision tree's framework, which could cause instability.

3.2 Model Building using TensorFlow

With the help of TF techniques, anyone can put best practices for data processing, model control, process management, and model retraining into reality with the aid of the TensorFlow framework. Process technology is required to automate and supervise model training during the production of an item, a service, or a business routine. The working process is described in Figure 3.3. The TensorFlow program uses data sets that are classified as algorithmic nodes resembling networks [11]. When the edges connecting the points in a graph represent multidimensional vectors or matrices, tensors are produced. since TensorFlow programs use generalized intermediate calculation results in their data flow architecture.

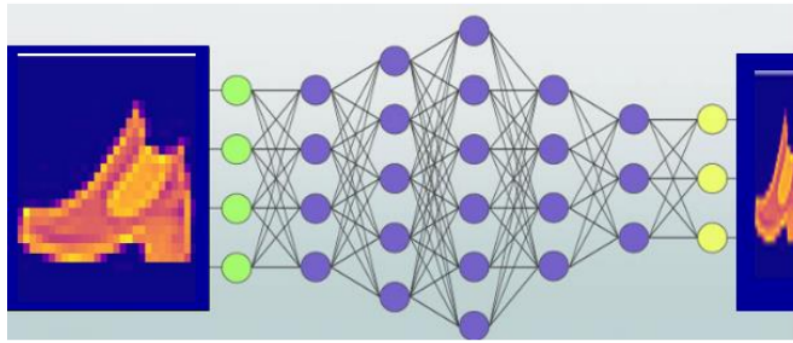


Figure 3.3 Working process of TensorFlow

The advantage of TF is it is an open-source platform. Also, TensorFlow is not limited to one specific device. It works as efficiently on a cellular device as it works on any other complex machine. TensorFlow has better data visualization power than any other available library for visualization techniques is TensorFlow. But it also has some disadvantages, and no windows support is available in this project. It takes more time to execute the model and the dependency also more.

3.3 Evaluation Matrices

Evaluation matrices help for finding the accuracy after executing the model. The above section already mentions that this dataset is a classification problem. So that the author considers only classification reports because they give the entire information about the outcome result. With the help of a classification report, anyone can easily find out the percentages of precision, recall, accuracy, and so on.

4 Result and Discussion

4.1 Data Description

MNIST Data is an image dataset it is a large database of Fashion. Also, it is the most popular dataset for training the model of the image processing machine. According to the project, there are two datasets available. One dataset has 60000 rows and 785 columns, and another dataset has 10000 rows and 785 columns. And all the column values indicate the pixel values of image data. Also, this dataset has one target column ("Label") and in this column 10 types ("0 to 9") of classes are present. So based on the target, this proposed problem comes under the classification techniques.

4.2 Import Packages

Library or packages played a vital role in every data science technique. So, for this reason, the author used some libraries to execute the project as well as handle the data in a better way. The author mentioned all the libraries which are already used in this project, this kind of information is provided by Listing 4.1.

```
import numpy as np # it helps for solving all the mathematical problems.
import pandas as pd # it helps for data manipulation.
import matplotlib.pyplot as plt # for visualization purpose this libraries is used.
import seaborn as sns # it also helps for visualiza the data but for better representation programmer used this libraries.
import matplotlib.image as mpimg # it helps for vizuliza the image data.
from sklearn.model_selection import train_test_split # for spliting data this library is used in this project.
from sklearn.metrics import confusion_matrix, classification_report # it helps for shows all the result.
from tensorflow.keras.utils import to_categorical# for Learn the categorical data.
from tensorflow.keras.models import Sequential # it helps for create the sequential model.
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout # for called the model this library is used
```

Listing 4.1 Provide the information about required Libraries.

NumPy helps for solving all the mathematical problems and at the same time, Pandas helps to load the dataset as well as manipulate data. For Visualization purposes, the author used Matplot and Seaborn because both libraries graphically provide valuable information. To visualize the image data the author used mpimg which comes under Matplotlib. Before modeling one of the important things is to divide the data, for this reason, the author also used here sklearn library and all the evaluation matrices also come under this library.

4.3 Data Loading

All the data science model wants some data to teach the model. Before starting any operation, the author loads two datasets and the Listing 4.2 shows all the information about the data load.

```
train = pd.read_csv("fashion-mnist_train.csv") # with the help of pandas library here the author Load the train data.
test = pd.read_csv("fashion-mnist_test.csv")# test dataset are Loaded with the help of pandas libraries.
```

Listing 4.2 Process of Data Loading

Here the author mentions the first dataset of this project. According to Figure 4.1, the total number of attributes is 42000 and the total number of features is 785. One based on 785 one is the output comes which is having 10 types of classes. Another thing is remaining all the columns have to indicate pixel values of the image data.

```
train # display the train dataset.
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782
0	2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	6	0	0	0	0	0	0	0	5	0	...	0	0	0	30	43	0	0	0
3	0	0	0	0	1	2	0	0	0	0	...	3	0	0	0	0	1	0	0
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59995	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
59996	1	0	0	0	0	0	0	0	0	0	...	73	0	0	0	0	0	0	0
59997	8	0	0	0	0	0	0	0	0	0	...	160	162	163	135	94	0	0	0
59998	8	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
59999	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

60000 rows × 785 columns

Figure 4.1 Data information about the First dataset.

Similarly, this project also has more than one dataset. This dataset has 28000 attributes and 784 features. This means the features are equal in both datasets according to Figure 4.2.

```
test # it shows the information about test data.
```

	label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782
0	0	0	0	0	0	0	0	0	9	8	...	103	87	56	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	...	34	0	0	0	0	0	0	0
2	2	0	0	0	0	0	0	14	53	99	...	0	0	0	0	63	53	31	0
3	2	0	0	0	0	0	0	0	0	0	...	137	126	140	0	133	224	222	56
4	3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	0	0	0	0	0	0	0	0	0	0	...	32	23	14	20	0	0	1	0
9996	6	0	0	0	0	0	0	0	0	0	...	0	0	0	2	52	23	28	0
9997	8	0	0	0	0	0	0	0	0	0	...	175	172	172	182	199	222	42	0
9998	8	0	1	3	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0
9999	1	0	0	0	0	0	0	0	140	119	...	111	95	75	44	1	0	0	0

10000 rows × 785 columns

Figure 4.2 Data information in the Second dataset.

4.4 Data Preprocessing

Before building the model one of the important things is to process the entire dataset properly. For this reason, the author used some functions for checking whether any missing values are present or not and how many maximum and minimum unique values are present in the entire columns. Also, the author finds if any duplicate values are present or not, and so on. Here the author considers only one dataset for executing the project.

```
train.isna().sum().sum(),test.isna().sum().sum() # find any null values are present or not
(0, 0)
```

Figure 4.3 Check how many nan values are present in both datasets.

There are no nan values present according to Figure 4.3.

```
train.nunique().min(), train.nunique().max() # find the maximum and minimum unique values.
(8, 256)
```

Figure 4.4 Find the minimum and maximum unique values in this dataset.

1 is the minimum unique value present in this dataset and at the same time, 256 is the highest unique value according to Figure 4.4.

```
train.duplicated().sum() # check any duplicate available or not
43
```

Figure 4.5 Find how many duplicate values are present in this dataset.

There are 43 duplicate values present according to Figure 4.5.

```
train["label"].isnull().sum()
0
```

Figure 4.6 Check whether any Null values are present or not.

No missing values are present in the target column according to Figure 4.6.

Figure 4.7 Display the Total number of unique values of this outcome column.

Here the author finds out the total unique values of all the target classes. According to Figure 4.7, the author finds that class 1 has the highest number of unique values, and class 5 has the lowest number of unique values.

```
train.columns # check all the name of the columns.
Index(['label', 'pixel0', 'pixel1', 'pixel2', 'pixel3', 'pixel4', 'pixel5',
      'pixel6', 'pixel7', 'pixel8',
      ...,
      'pixel774', 'pixel775', 'pixel776', 'pixel777', 'pixel778', 'pixel779',
      'pixel780', 'pixel781', 'pixel782', 'pixel783'],
      dtype='object', length=785)
```

Figure 4.8 Shows all the column names of this project.

According to Figure 4.8, the author shows all the column names.

4.5 Data Visualization

Data visualization is an important topic of every data science project. Because with the help of visualization, many hidden pieces of information are easy to point out. For this reason, the author used some visualization techniques for getting more valuable information.

```
plt.figure(figsize=(10, 8)) # set the figure size
sns.countplot(train['label'], palette='viridis') # with the help of seaborn, here the author used count plot
plt.ylim(3700, ) # adjust the y axis values for better visulaization
plt.show() # Display the graph.
```

C:\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 misinterpretation.
warnings.warn()

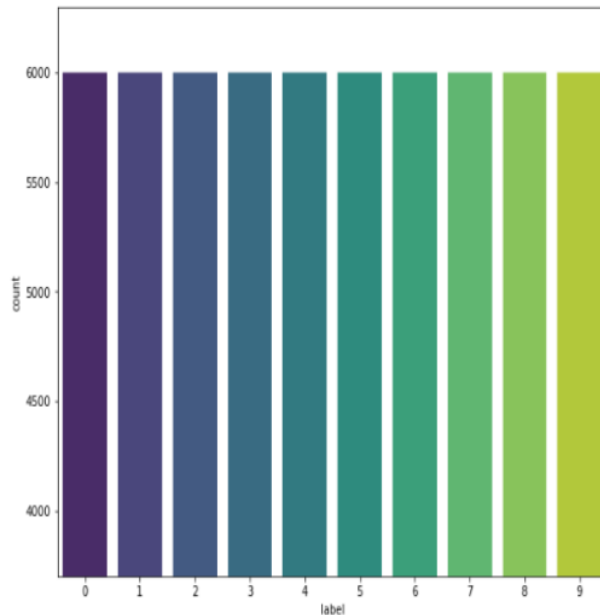


Figure 4.9 Represent the information about the target variable graphically.

The target variable is one of the important factors of this project so it must be important to explore this column in a better way. According to Figure 4.9, all the category are equal values and category 5 has fewer values. Similarly, Figure 4.10 shows the same image data of this project.

```
# first train images with labels
fig, ax = plt.subplots(figsize=(18, 8))
for ind, row in train.iloc[:5, :].iterrows():
    plt.subplot(2, 4, ind+1)
    plt.title(row[0])
    img = row.to_numpy()[1:].reshape(28, 28)
    fig.suptitle('Train images', fontsize=24)
    plt.axis('off')
    plt.imshow(img, cmap='plasma')
```

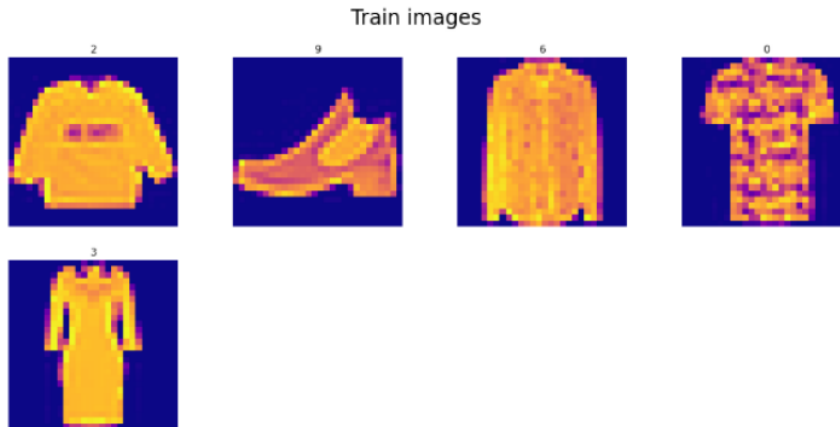


Figure 4.10 Display All the input images.

4.6 Model Creation

Before starting the running model, some libraries are required. The author imports some libraries to run the model for finding better accuracy. According to the target column, it is a multiclass classification problem, so the author imports random forest and decision tree here because both algorithms easily handle the multi-class problem. To display the result the author used a classification report which comes under the Sklearn packages and all the information is available in Listing 4.3.

```
from sklearn.ensemble import RandomForestClassifier # import Random Forest Classifier for predict the model.
from sklearn.model_selection import train_test_split # split the data.
from sklearn.metrics import classification_report, confusion_matrix # this library helps for evaluate the model.
#To show graphs within the notebook
%matplotlib inline
```

Listing 4.3 shows all the required libraries for running the model.

```
y = train.pop("label") # store target variable
X = train # store entire dataset with out target column.
```

Listing 4.4 Dividing the dataset.

Data splitting is needed before running the model. Here target columns are stored in one variable and the rest of the variables are stored in another variable. According to Listing 4.4, the target variable is stored in y and the rest of the features are stored in the variable X.

```

: #Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print(X_train.shape) # print shape of the X_train data.
print(X_test.shape) # print shape of the X_test data.

(42000, 784)
(18000, 784)

```

Figure 4.11 Display the shape of the train and test data.

In ML, before testing the model it must be important to properly train the model. So that reason the author divides the dataset and 70% data saved for training purposes and the remaining 30 % of the data stored for testing purposes. 29400 data are present in the train according to Figure 4.11 and 12600 data are stored in the test.

```

#training random Forest
rf=RandomForestClassifier(n_estimators=100) # called random forest classifier with 100 estimators.
rf.fit(X_train,y_train) # Fit dataset

```

```

RandomForestClassifier
RandomForestClassifier()

```

Figure 4.12 Called random forest classifier.

It is a multiclass classification problem so the author used RF here. So, the author called random forest classifier and fit train data, and also the author mentions the estimator value 100 according to Figure 4.12.

```

pred=rf.predict(X_test)# predict the model.
print ("Classification Report")
print(classification_report(y_test, pred))
print ("Confusion Report")
print(confusion_matrix(y_test, pred))

```

Classification Report					
	precision	recall	f1-score	support	
0	0.82	0.86	0.84	1799	
1	1.00	0.96	0.98	1843	
2	0.79	0.84	0.81	1808	
3	0.88	0.92	0.90	1849	
4	0.78	0.83	0.80	1765	
5	0.97	0.96	0.96	1786	
6	0.74	0.60	0.66	1787	
7	0.94	0.94	0.94	1789	
8	0.96	0.97	0.96	1795	
9	0.94	0.96	0.95	1779	
accuracy			0.88	18000	
macro avg	0.88	0.88	0.88	18000	
weighted avg	0.88	0.88	0.88	18000	

```

Confusion Report
[[1549  0  26  57  6  2 142  0 17  0]
 [ 4 1772 10 40  4  0 11  0  2  0]
 [ 9  0 1510 10 190  0 79  0 10  0]
 [39  2  11 1708 58  0 31  0  0  0]
 [ 5  1 138  69 1470  0 75  0  7  0]
 [ 0  0  0  1  0 1709  0 51  6 19]
 [283  4 203 41 157  2 1066  0 31  0]
 [ 0  0  0  0  0  29  0 1679  2 79]
 [ 0  0  8  6  8  5 27  2 1737  2]
 [ 0  0  0  0  0 19  1  58  1 1700]]

```

Figure 4.13 Accuracy of the RF model.

According to Figure 4.13, the RF got 88% accuracy after executing the model.

```
from sklearn.tree import DecisionTreeClassifier # import decision tree for classified the outcomes.
dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)#called decision tree classifier $ fit the train data.
dtree_model
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=2)
```

Figure 4.14 Called Decision Tree Classifier.

Since there are multiple classes involved, the author used RF in this situation. The author used Figure 4.14 to match the train data to the decision tree classifier.

```
pred=dtree_model.predict(X_test)# predict the model with the help of decision tree.
print ("Classification Report")
print(classification_report(y_test, pred))# shows the classification report.
print ("Confusion Report")
print(confusion_matrix(y_test, pred))
```

```
Classification Report
precision    recall  f1-score   support

0           0.00      0.00      0.00      1799
1           0.92      0.83      0.87      1843
2           0.00      0.00      0.00      1808
3           0.00      0.00      0.00      1849
4           0.17      0.99      0.29      1765
5           0.00      0.00      0.00      1786
6           0.00      0.00      0.00      1787
7           0.51      0.89      0.65      1789
8           0.00      0.00      0.00      1795
9           0.56      0.86      0.67      1779

 accuracy          0.36      18000
 macro avg         0.22      18000
weighted avg         0.22      18000

Confusion Report
[[ 0 20 0 0 1735 0 0 15 0 29]
 [ 0 1529 0 0 302 0 0 6 0 6]
 [ 0 20 0 0 1762 0 0 8 0 18]
 [ 0 10 0 0 1792 0 0 12 0 35]
 [ 0 7 0 0 1747 0 0 5 0 6]
 [ 0 36 0 0 65 0 0 1258 0 427]
 [ 0 20 0 0 1742 0 0 12 0 13]
 [ 0 0 0 0 2 0 0 1597 0 190]
 [ 0 10 0 0 1229 0 0 65 0 491]
 [ 0 1 0 0 100 0 0 156 0 1522]]
```

Figure 4.15 Accuracy of DT

36 % is the accuracy of DT based on Figure 4.15.

For finding better accuracy the author used some advanced techniques here. For this reason, the author imports ANN and split the data, these kinds of information are available in Listing 4.5 and 4.6.

```
# Deep Learning Libraries
import tensorflow as tf
from tensorflow.keras import layers
import keras
from keras.models import Sequential
from keras.layers.core import Dense,Activation,Dropout
from keras.datasets import mnist
#from keras.utils import to_categorical
from tensorflow.keras.utils import to_categorical# for learn the categorical data.
from tensorflow.keras.models import Sequential # it helps for create the sequential model.
from keras.wrappers.scikit_learn import KerasClassifier
```

Listing 4.5 import all the libraries for deep learning techniques.

```
X_train_fashion = train.drop('label',axis = 1) #drop target column and store rest of the columns from train data.
y_train_fashion = train['label'] # store target columns
X_test_fashion = test.drop('label',axis = 1)#drop target column and store rest of the columns from test data.
y_test_fashion = test['label'] # target column saved in a variable.
```

Listing 4.6 split data into train and test

```
#Reshaping the MNIST dataset
x_train_reshape = X_train_fashion.values.reshape(-1,28,28)# reshape x train data.
x_test_reshape = X_test_fashion.values.reshape(-1,28,28) # reshape x test data.

col_names = ['T-shirt/top','Trouser','Pullover','Dress','Shirt','sandal','Coat','Ankle boot','Bag','shoes'] # item names

#Visualizing the data images
plt.figure(figsize=(10,10)) # fixed figure size
for i in range(15):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train_reshape[i], cmap='Blues') # shows the images
    plt.xlabel(col_names[y_train_fashion[i]])
plt.show() # display plot
```



Figure 4.16 represent ten different classes after resampling.

```
#Encoding Labels of Fashion MNIST
y_train_fashion = to_categorical(y_train_fashion, num_classes=10) # encode train dataset.
y_test_fashion = to_categorical(y_test_fashion, num_classes=10) # encode test dataset.
```

Listing 4.7 encode the target variable

```

#Create function using neural network
model2 = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(784,)), # Input layer of the model
    layers.Dropout(0.3), # Adding dropout to avoid overfitting
    layers.BatchNormalization(),
    layers.Dense(24, activation='relu'), # First Hidden Layer
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(24, activation='relu'), # second hidden layer of the model
    layers.Dropout(0.3),
    layers.BatchNormalization(),
    layers.Dense(10, activation='softmax'), # Output Layer of the model
])

#Compiling the model using user define function
model2.compile(loss="categorical_crossentropy", # Model compiling using categorical cross entropy because multiclass classification
               optimizer="adam",
               metrics = ['accuracy']) # Considering accuracy metrix to evaluate model performance

#Fit the model
model2.fit(X_train_fashion, y_train_fashion, batch_size=100, epochs=30) # Model fottting means training part using train data

```

Listing 4.8 execute neural network for predict the result.

Here the author used one function to execute the model. According to Listing 4.8, the author used relu and SoftMax as activation functions and the author used 2 hidden layers. Also, the author used adam as an optimizer of this code.

```

600/600 [=====] - 2s 3ms/step - loss: 1.1014 - accuracy: 0.6241
Epoch 2/30
600/600 [=====] - 2s 3ms/step - loss: 0.7352 - accuracy: 0.7507
Epoch 3/30
600/600 [=====] - 2s 3ms/step - loss: 0.6765 - accuracy: 0.7727
Epoch 4/30
600/600 [=====] - 2s 3ms/step - loss: 0.6392 - accuracy: 0.7862
Epoch 5/30
600/600 [=====] - 2s 3ms/step - loss: 0.6232 - accuracy: 0.7908
Epoch 6/30
600/600 [=====] - 2s 3ms/step - loss: 0.6078 - accuracy: 0.7976
Epoch 7/30
600/600 [=====] - 2s 3ms/step - loss: 0.5913 - accuracy: 0.8022
Epoch 8/30
600/600 [=====] - 2s 3ms/step - loss: 0.5824 - accuracy: 0.8038
Epoch 9/30
600/600 [=====] - 2s 3ms/step - loss: 0.5701 - accuracy: 0.8086
Epoch 10/30
600/600 [=====] - 2s 3ms/step - loss: 0.5648 - accuracy: 0.8109
Epoch 11/30
600/600 [=====] - 2s 3ms/step - loss: 0.5519 - accuracy: 0.8144
Epoch 12/30
600/600 [=====] - 2s 3ms/step - loss: 0.5541 - accuracy: 0.8127
Epoch 13/30
600/600 [=====] - 2s 3ms/step - loss: 0.5495 - accuracy: 0.8152
Epoch 14/30
600/600 [=====] - 2s 3ms/step - loss: 0.5499 - accuracy: 0.8151
Epoch 15/30
600/600 [=====] - 2s 3ms/step - loss: 0.5402 - accuracy: 0.8188
Epoch 16/30
600/600 [=====] - 2s 3ms/step - loss: 0.5213 - accuracy: 0.8262
Epoch 17/30
600/600 [=====] - 2s 3ms/step - loss: 0.5234 - accuracy: 0.8247
Epoch 18/30
600/600 [=====] - 2s 3ms/step - loss: 0.5235 - accuracy: 0.8248
Epoch 19/30
600/600 [=====] - 2s 3ms/step - loss: 0.5222 - accuracy: 0.8248
Epoch 20/30
600/600 [=====] - 2s 3ms/step - loss: 0.5173 - accuracy: 0.8260
Epoch 21/30
600/600 [=====] - 2s 3ms/step - loss: 0.5194 - accuracy: 0.8268
Epoch 22/30
600/600 [=====] - 2s 3ms/step - loss: 0.5156 - accuracy: 0.8274
Epoch 23/30
600/600 [=====] - 2s 3ms/step - loss: 0.5152 - accuracy: 0.8260
Epoch 24/30
600/600 [=====] - 2s 3ms/step - loss: 0.5091 - accuracy: 0.8304
Epoch 25/30
600/600 [=====] - 2s 3ms/step - loss: 0.5100 - accuracy: 0.8308
Epoch 26/30
600/600 [=====] - 2s 3ms/step - loss: 0.5071 - accuracy: 0.8296
Epoch 27/30
600/600 [=====] - 2s 3ms/step - loss: 0.5003 - accuracy: 0.8312
Epoch 28/30
600/600 [=====] - 2s 3ms/step - loss: 0.5111 - accuracy: 0.8298
Epoch 29/30
600/600 [=====] - 2s 3ms/step - loss: 0.5038 - accuracy: 0.8310
Epoch 30/30
600/600 [=====] - 2s 3ms/step - loss: 0.4974 - accuracy: 0.8341

```

Figure 4.17 shows the output of the model.

```
#Evaluating fashion MNIST using test data for finding the accuracy.  
test_loss_fashion, test_acc_fashion = model2.evaluate(X_test_fashion, y_test_fashion) # evaluate the model  
313/313 [=====] - 1s 1ms/step - loss: 0.3827 - accuracy: 0.8721
```

Figure 4.18 test the model for getting the result.

4.7 Final Result

After completing the execution of this proposed project, the author got 87% accuracy from ANN techniques. So. It indicates that with the help of ANN, the author easily finalizes the model.

5 Conclusion

5.1 Conclusion

The MNIST dataset is an image-based dataset, all the Fashion related images are available in this dataset. In this project, the author used ML and DL techniques to predict the model. According to the model, the author got 87% accuracy which is the highest accuracy of this model. Also, the author used machine learning algorithms but using ML the highest accuracy is 94% from RF, and it also has good accuracy for selecting the model. Based on the highest accuracy the author selects the DL techniques for this project.

5.2 Recommendation and Future Scope

Here author applies only a few ML techniques and TF techniques to get the desired output. But it is recommended that instead of ML techniques if Deep Learning techniques such as CNN, DNN etc. are used then the model performance can be increased and the results also be more accurate. Here author only applies RF and DT algorithms used and it is known that naïve bayes is the best suitable for classification but here author didn't use them because it takes more time to train the model. So, to get the more accurate result it is recommended to use NB techniques with a high processor system.

6 Reference

- [1] Vijayaraj, A., Vasanth Raj, R. Jebakumar, P. Gururama Senthilvel, N. Kumar, R. Suresh Kumar, and R. Dhanagopal. "Deep Learning Image Classification for Fashion Design." *Wireless Communications and Mobile Computing* 2022 (2022).
- [2] LEITHARDT, VALDERI. "Classifying garments from fashion-MNIST dataset through CNNs." *Advances in Science, Technology and Engineering Systems Journal* 6, no. 1 (2021): 989-994.
- [3] Greeshma, K. V., and K. Sreekumar. "Fashion-MNIST classification based on HOG feature descriptor using SVM." *International Journal of Innovative Technology and Exploring Engineering* 8, no. 5 (2019): 960-962.
- [4] Abu, Mohd Azlan, Nurul Hazirah Indra, A. H. A. Rahman, Nor Amalia Sapiee, and Izanoordina Ahmad. "A study on Image Classification based on Deep Learning and Tensorflow." *International Journal of Engineering Research and Technology* 12, no. 4 (2019): 563-569.
- [5] Schulz, Marc-Andre, BT Thomas Yeo, Joshua T. Vogelstein, Janaina Mourao-Miranada, Jakob N. Kather, Konrad Kording, Blake Richards, and Danilo Bzdok. "Different scaling of linear models and deep learning in UKBiobank brain images versus machine-learning datasets." *Nature communications* 11, no. 1 (2020): 4238.
- [6] Kadam, Shivam S., Amol C. Adamuthe, and Ashwini B. Patil. "CNN model for image classification on MNIST and fashion-MNIST dataset." *Journal of scientific research* 64, no. 2 (2020): 374-384.
- [7] Kayed, Mohammed, Ahmed Anter, and Hadeer Mohamed. "Classification of garments from fashion MNIST dataset using CNN LeNet-5 architecture." In *2020 international conference on innovative trends in communication and computer engineering (ITCE)*, pp. 238-243. IEEE, 2020.
- [8] LEITHARDT, VALDERI. "Classifying garments from fashion-MNIST dataset through CNNs." *Advances in Science, Technology and Engineering Systems Journal* 6, no. 1 (2021): 989-994.
- [9] Arora, Nisha, and Pankaj Deep Kaur. "A Bolasso based consistent feature selection enabled random forest classification algorithm: An application to credit risk assessment." *Applied Soft Computing* 86 (2020): 105936.

- [10] Tonner, Peter D., Abe Pressman, and David Ross. "Interpretable modeling of genotype–phenotype landscapes with state-of-the-art predictive power." *Proceedings of the National Academy of Sciences* 119, no. 26 (2022): e2114021119.
- [11] Mardanirad, Sajjad, David A. Wood, and Hassan Zakeri. "The application of deep learning algorithms to classify subsurface drilling lost circulation severity in large oil field datasets." *SN Applied Sciences* 3, no. 9 (2021): 785.

7 Appendix

```
import numpy as np # it helps for solving all the mathematical problems.

import pandas as pd # it helps for data manipulation.

import matplotlib.pyplot as plt # for visualization purpose this libraries is used.

import seaborn as sns # it also helps for visualiza the data but for better representation
programmer used thiis libraries.

import matplotlib.image as mpimg # it helps for visuliza the image data.

from sklearn.model_selection import train_test_split # for spliting data this library is used in
this project.

from sklearn.metrics import confusion_matrix, classification_report # it helps for shows all the
result.

from tensorflow.keras.utils import to_categorical# for learn the categorical data.

from tensorflow.keras.models import Sequential # it helps for create the sequential model.

from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout # for
called the model this library is used

train = pd.read_csv("fashion-mnist_train.csv") # with the help of pandas library here the author
load the train data.

test = pd.read_csv("fashion-mnist_test.csv")# test dataset are loaded with the help of pandas
libraries.

train # display the train dataset.

test # it shows the information about test data.

train.isna().sum().sum(),test.isna().sum().sum() # find any null values are present or not

train.nunique().min() , train.nunique().max() # find the maximum and minimum unique values.

train.duplicated().sum() # check any duplicate avaiable or not

train["label"].isnull().sum()

train["label"].value_counts().sort_values # find the unique values of the target variable.
```

```

train.columns # check all the name of the columns.

plt.figure(figsize=(10, 8)) # set the figure size

sns.countplot(train['label'], palette='viridis') # with the help of seaborn, here the author used
count plot

plt.ylim(3700, ) # adjust the y axis values for better visulaization

plt.show() # Display the graph.

len(train), len(test) # check length of the both the data set.

# first train images with labels

fig, ax = plt.subplots(figsize=(18, 8))

for ind, row in train.iloc[:5, :].iterrows():

    plt.subplot(2, 4, ind+1)

    plt.title(row[0])

    img = row.to_numpy()[1:].reshape(28, 28)

    fig.suptitle('Train images', fontsize=24)

    plt.axis('off')

    plt.imshow(img, cmap='plasma')

from sklearn.ensemble import RandomForestClassifier # import Random Forest Classifier for
predict the model.

from sklearn.model_selection import train_test_split # split the data.

from sklearn.metrics import classification_report, confusion_matrix # this library helps for
evaluate the model.

#To show graphs within the notebook

%matplotlib inline

y = train.pop("label") # store target variable

X = train # shore entire dataset with out target column.

```

```

#Split the dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

print(X_train.shape) # print shape of the X_train data.

print(X_test.shape) # print shape of the X_test data.

#training random Forest

rf=RandomForestClassifier(n_estimators=100) # called random forest classifier with 100
estimators.

rf.fit(X_train,y_train) # Fit dataset

pred=rf.predict(X_test)# predict the model.

print ("Classification Report")

print(classification_report(y_test, pred))

print ("Confusion Report")

print(confusion_matrix(y_test, pred))

from sklearn.tree import DecisionTreeClassifier # import decision tree for classified the
outcomes.

dtree_model = DecisionTreeClassifier(max_depth = 2).fit(X_train, y_train)#called decision
tree classifier $ fit the train data.

dtree_model

pred=dtree_model.predict(X_test)# predict the model with the help of decision tree.

print ("Classification Report")

print(classification_report(y_test, pred))# shows the classification report.

print ("Confusion Report")

print(confusion_matrix(y_test, pred))

import tensorflow as tf # import tensorflow methods.

```

```

train = pd.read_csv("fashion-mnist_train.csv") # with the help of pandas library here the author
load the train data.

test = pd.read_csv("fashion-mnist_test.csv")

test = test.drop(['label'], axis=1)

# split into image and labels and convert to numpy array

X = train.iloc[:, 1:].to_numpy()

y = train['label'].to_numpy()

# test dataset

test = test.loc[:, :].to_numpy()

X = X / 255.0

test = test / 255.0

test.shape

print(X.shape)# shows shape of the X variable

print(test.shape)# shows shape of the test variable

X = X.reshape(-1,28,28,1) # reshape the X data.

test = test.reshape(-1,28,28,1)# reshape the test data

print(X.shape)# shows shape of the X variable after reshaping.

print(test.shape)# shows shape of the X variable after reshaping.

print(y.shape)

print(y[0])

y_enc = to_categorical(y, num_classes = 10)

print(y_enc.shape)

print(y_enc[0])

# split the data

random_seed = 2

```

```

X_train, X_val, y_train_enc, y_val_enc = train_test_split(X, y_enc, test_size=0.3)

for i in [X_train, y_train_enc, X_val, y_val_enc]:

    print(i.shape)

g = plt.imshow(X_train[0][:,:,0]) # shows the fashion image
print(y_train_enc[0])

g = plt.imshow(X_train[4][:,:,0]) # shows the fashion image
print(y_train_enc[4])

# Execute the model with the help of Keras.

model = tf.keras.Sequential([

    tf.keras.layers.Conv2D(16, kernel_size=3, activation='relu'),

    tf.keras.layers.MaxPooling2D(pool_size=2),

    tf.keras.layers.Conv2D(32, kernel_size=3, activation='relu'),

    tf.keras.layers.MaxPooling2D(pool_size=2),

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(64, activation='relu'),

    tf.keras.layers.Dense(10, activation='softmax')

])

model.compile(loss='categorical_crossentropy',

              optimizer='adam',

              metrics='accuracy') # Compile the model.

result = model.fit(X_train, y_train_enc,

                  validation_data=(X_val, y_val_enc),

                  epochs=10, batch_size=8) # fit the model for find the accuracy of the project

```

8 Video Link

<https://www.youtube.com/watch?v=7f4BVz5zgAA>

Classification using ML methods and Neural Networks

GRADEMARK REPORT

FINAL GRADE

48/100

GENERAL COMMENTS

Instructor

Well structured document. An abstract would help here. Nice to see a list of abbreviations. Well done.

Your lit review covers related work well and includes relevant concepts and theory. Well done. Refs used appropriately, again, good.

Your experiment/method is clear and the results of the fashion dataset trained models as expected. Try to further justify your parameter choices here and present results in the form of bar graphs for comparison along with tables.

There is a big problem unfortunately, here. YOU DID NOT ATTEMPT THE SECOND PART OF THE ASSESSMENT - YOU DID NOT TRAIN AND EVALUATE YOUR MODELS USING THE CIFAR 10 DATASET. THIS IS A SERIOUS LIMITATION AND HENCE THE LOWER MARK.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26

PAGE 27

PAGE 28

PAGE 29

GRADING FORM: SMU GENERIC_GJ

GOWRI PAMARTHI

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.