An Industry Oriented Mini Project (CS704PC)

on

"CLASSIFICATION OF GARMENTS USING FASHION MNIST DATASET"

Submitted
in the partial fulfillment of the requirements for
the award of the degree of

Bachelor of Technology

in

Computer Science Engineering

by

Mr. KAMBHAM KOUSHIK

(19261A05E2)

Under the guidance of

Dr. K. Sreekala

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

GANDIPET, HYDERABAD-500 075, INDIA

2022-2023

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

GANDIPET, HYDERABAD-500 075, Telangana.

CERTIFICATE

This is to certify that the industry oriented MINI project (CS704PC) entitled "**CLASSIFICATION OF GARMENTS USING FASHION MNIST DATASET",** is being submitted by **Mr. Kambham Koushik** bearing **Roll No: 19261A05E2** in partial fulfillment for the award of degree of **B.Tech** in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out under the guidance of **Dr. K. Sreekala, Assistant Professor, Department of CSE.**

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guide
Dr. K. Sreekala
Assistant Professor, Dept. of CSE

Head of the Department
Dr. C.R.K Reddy
Professor, Dept.of CSE

External Examiner

DECLARATION

This is to certify that the work reported in this project titled "**CLASSIFICATION OF GARMENTS USING FASHION MNIST DATASET"** is a record of work done by me in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the work done entirely by me and not copied from any other source.

KAMBHAM KOUSHIK
19261A05E2

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the project.

I would like to express my sincere thanks to **Prof. G. Chandra Mohan Reddy, Principal MGIT**, for providing the working facilities in college.

I wish to express my sincere thanks and gratitude to **Dr. C R K Reddy, Professor and HOD**, Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to **Dr. M. Sreevani, Associate Professor, Dr. K. Mahesh Kumar, Associate Professor, Department of CSE, MGIT**, Mini Project Coordinators for their encouragement and support throughout the project.

I am extremely thankful and indebted to my internal guide **Dr. K. Sreekala, Assistant Professor**, **Department of CSE**, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of CSE Department who helped me directly or indirectly, for completing this project.

KAMBHAM KOUSHIK
19261A05E2

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

MNIST	Modified National Institute of Standards and Technology
SGD	Stochastic Gradient Descent
XGB	Extreme Gradient Boost
CNN	Convolutional Neural network
SVM	Support Vector Machine
LSTM	Long Short-Term Memory

LIST OF TABLES

ABSTRACT

Online fashion market is constantly growing and fashion websites are also getting more data from different brands. Due to this, classification of different garments has became very tough for many websites. To solve this problem, An algorithm with high accuracy is required in which it can identify garments that can help companies in the clothing sales sector to understand the profile of potential buyers and focus on sales targeting specific niches, as well as developing campaigns based on the taste of customers and improve user experience.

This project is aimed to find the best model with highest accuracy and precision results. Models like Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Stochastic Gradient Descent, XGBoost and sequential model are used in this project to classify the garments. To train and test these models, the Fashion MNIST dataset is used. This dataset is published in Zolando's website.

These models are trained to classify the 60,000 small square 28×28 pixels grayscale images in the Fashion MNIST Dataset into the 9 different classes.

After training and testing these models, Accuracy, precision and some other performance measures are calculated and compared with one another. Among all the models which are used here, the model which shows best performance is suggested to the fashion website.

1. INTRODUCTION

The fashion market has changed dramatically over the last 30 years, resulting in an evolution in that industry. Understanding customer tastes and better-directing sales are the way to increase profit. The rise of internet business lets people buy their clothes through websites, faster and easier. The introduction of methods to improve user's experience when searching for items in these platforms is decisive. Classifying clothes is part of the broad task of classifying scenes. The automatic generation of image labels that describe those products can alleviate human annotators' workload. This kind of information may also help labeling scenes and better understanding users' tastes, culture, and financial status.

In the Fashion – MNIST data set based on images from Zalando, which is the Europe's largest online fashion platform. Fashion MNIST has 60,000 products with 28x28 pixel grey scale images divided into 10 categories: t-shirt, trouser, pullover, dress, coat, sandals, shirt, sneaker, bags and ankle boots.

The evaluated implementations (from scikit-learn) in this project were: Decision Tree, Logistic Regression, Random Forest, Stochastic Gradient Descent, XGBoost and sequential model. These models are trained using the Fashion MNIST dataset and performance measures are calculated for each of the models. The model which gives best performance measures is tested with the testing dataset. Among all the models sequential model has the best performance measures when compared to the remaining models.

1.1 Problem Definition:

Classification of garments has become a very tough task for many websites due to large data and many different images. Even though there are many methods in classifying the clothes there is some misclassification and output is not as expected i.e., when you search for shirt in any website then the website will suggest some extra outfits like pant, shoes which are not required for us and irritate us sometimes. To solve this problem models like Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Stochastic Gradient Descent, XGBoost and sequential model are used and fashion MNIST dataset. The main aim of this project is to classify the clothes into different classes using Fashion MNIST dataset. So, in this project machine learning algorithms such as sequential model and logistic regression are used to get better results.

1.2 Existing System:

There are good number of models exists for classifying the images, but the data used here is not up to the mark and don't provide required outcome due to the richness of cloth properties and the high depth of clothes. Existing model is built using LeNet-5 Model. Even though, they use this model in wide range of domains but there are some challenges in training a Convolutional Neural Network and they are overfitting and accuracy is not high.

Problems in Existing System:

- Accuracy is not equal for each class of classification
- Problem in classification due to richness of clothes properties
- Moderate Error rate
- Overfitting

1.3 Proposed System:

The proposed system (Classification of garments using fashion MNST dataset) is implemented using python and Machine Learning Algorithms in google colab notebook platform. Different algorithms are used to classify the dataset and performance measures are calculated for each model. Here the train and test data is separately published by zolando website and accuracy is calculated making sure that this method is helpful in making classifying the garments and overcomes the problems of existing system.

1.4 Requirements Specifications:

Requirement Specifications describe the articraft of Software Requirements and Hardware Requirements used in this project.

1.4.1 Software Requirements:

Operating System : Windows 8 or above

IDE : Google colab notebook

Language : Python 3.5 and above

Packages required : numpy, pandas, sklearn , matplotlib, keras

1.4.2 Hardware Requirements:

RAM : 4GB or more

Hard Disk : 1TB

Processor : Any Intel/Ryzen process

2. LITERATURE SURVEY

The following literature reviews discusses about various methods and models used for classification of garments.

"Cloth Classifying from Fashion-MNIST Dataset" by Anita Maria Rocha, V.Q. Leithardt, L. Rodrigo, S. Correia. They proposed SVM, Random Forest Classifier and drop-out technique for Cloth Classification. The main goal is to compare those results with the original one, providing future research to be able to easily choose the most suitable classification method. This model demonstrated a good performance measures and accuracy from 89.7% to 99.1%. The major drawback of this paper is, Classification problem, due to the richness of clothes properties and high depth of cloth categorization.

"**Classification of Garments Using CNN LeNet-5 Model**" by Mohammed Kayed, A. Anter, H. Mohamed. LeNet-5 outperforms both classical CNN and state of the art model. The demerit of this model is it cannot deal with challenges in training a Convolutional Neural Network and overfitting is also cannot be handled.

"**Fashion-MNIST Classification Based on HOG Feature Descriptor Using SVM**" by Greeshma KV, Sreekumar K. In this paper they classified the dataset based on HOG feature descriptor. They used Support Vector Machine (SVM) classifier algorithm. Min reason for using SVM is it is the best method to train the images. But Accuracy is not clear for each class of classification.

"**Image Classification of Fashion-MNIST Dataset Using Long Short-Term Memory Networks**" by Shuning Shen in 2018. It discusses the LSTMs model, Network pruning and Training Pattern Reduction. Computational cost of this model is low. Disadvantage of this paper is Performance of training pattern sets is worse when number of training patterns is reducing.

"**ImageNet Classification with Deep Convolutional Neural Networks**" by Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. Described about Local response normalization and also about training pattern reduction. In this paper is Non-saturating neurons of conventional operation are used, so that training is done faster. Disadvantage is the models used in this paper achieve error rate more than 35%.

Table 2.1: Literature survey of Classification of garments using fashion MNIST dataset

S.NO	Author and Year of Publication	Title	Technology/ Algorithm Used	Advantages	Disadvantages
1	V.Q. Leithardt, L. Rodrigo, S. Correia 2021	Cloth Classifying from Fashion MNIST dataset	SVM and Random Forest Classifier, Drop-out technique	This method enhance accuracy from 89.7% (using SVM) to 99.1% (using cnn)	Classification problem, due to the richness of the clothes properties and the high depth of clothes categorization
2	M. Kayed, A. Anter, H. Mohamed 2020	Classification of Garments Using CNN LeNet-5 Architecture	CNN based LeNet-5 Model	LeNet-5 outperforms both classical CNN and state of the art model	Challenges in training a Convolutional Neural Network is overfitting
3	Greeshma K V, Sreekumar K 2019	Fashion-MNIST Classification Based on HOG Feature Descriptor	SVM(Support Vector Machine) Classifier	Multiicalss SVM is the best method to train the images	Accuracy is not equal for each class of classification
4	Shuning Shen 2018	Image Classification Using Long Short-Term Memory Networks	LSTMs model, , Network Pruning and Training Pattern Reduction	Computational cost of this model is low	Performance of training pattern sets is worse when number of training patterns is reducing
5	A. Krizhevsky, I. Sutskever, G.E. Hinton 2018	ImageNet Classification with Deep Convolutional Neural Networks	Local Response Normalization, Training on Multiple GPUs	Non-saturating neurons of convolution operation are used. By this training is done faster	These models achieve error rate of more than 35%.

3. DESIGN AND METHODOLOGY

Design is the foundation for creating any software or model. Software design is basically a mechanism of preparing a plan, a layout for structuring the code of software application. The design of the project helps us to keep in check of the following important points: Modularity, Maintainability, The Flow of functionality and performance and Portability, Tractability.

Methodology is a set of principles, tools and techniques that are used to plan, execute and manage projects. Moreover, Methodology refers to the overarching strategy and rationale of your research project. It involves studying the methods used in your field and the theories or principles behind them, in order to develop an approach that matches your objectives.

3.1 Architecture diagram of Classification of garments using fashion MNIST dataset

The Objective is to design a system which classifies the data into required items and different classes, this consists of using Fashion MNIST dataset and various algorithms. The approach comprises a sequence of steps for loading the dataset and classifying the dataset into 10 different classes and also standardizing the dataset, so that all the values of each feature (pixel) is in a small range (based on the standard deviation value). To get good prediction for some models we should consider a large dataset. The general work-flow of the program can be seen in Figure 3.1.1.

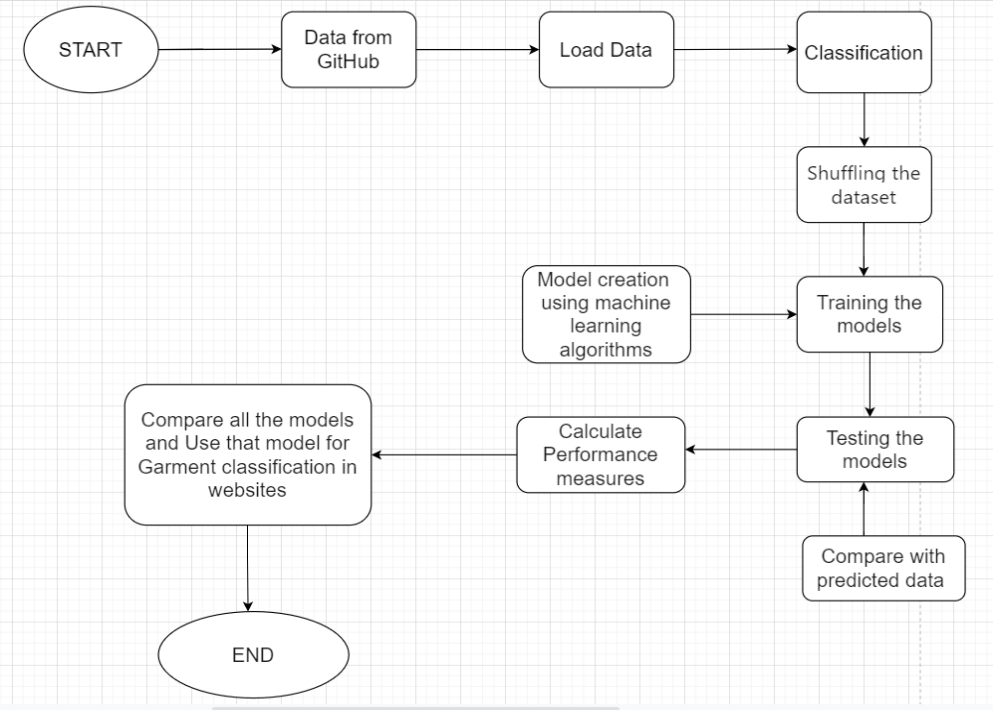

Fig. 3.1.1 General Work-flow of Classification of Garments using fashion MNIST dataset

As shown in the Figure above the data is retrieved from Github and then it is loaded to into the program. Then the data is classified into different classes. Dataset is the shuffled to make sure there is no bias in the dataset. By using the dataset we train the models and performance measures are calculated for these models then the best testing is performed on the model which has the best accuracy.

3.2 Dataset

The most important criteria for good classification is to have the accurate dataset. Fashion MNIST dataset is considered for classification of garments.

The dataset we have taken from Github - consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label.

The dataset obtained doesn't have any noisy and missing values. Also, all the attributes needed to get best results are acquired in this dataset. So, the classification is directly done by training the models with the data. Some pixel values out of 785 are showed in the below figure 3.2.

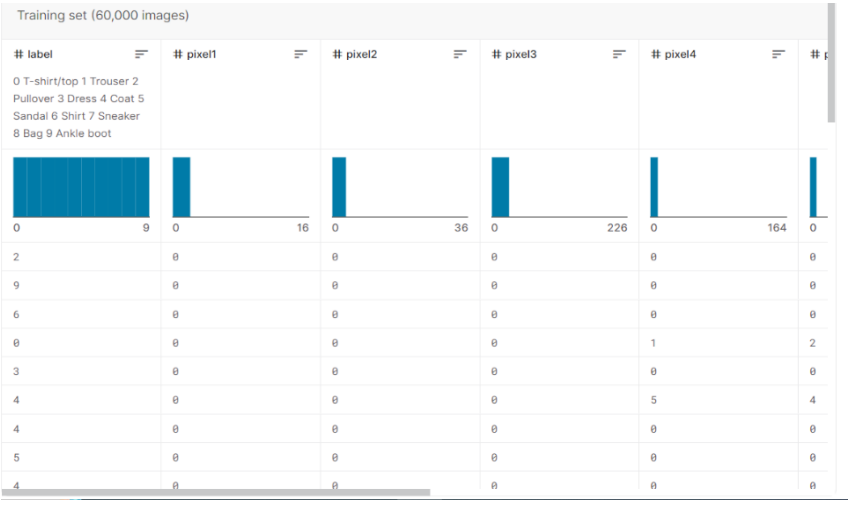

Fig. 3.2.1 Preview of Training dataset of 60,000 images

Fig. 3.2.2 Preview of Testing dataset of 10,000 images

3.3 Models

Various models are used in classifying the Garments from dataset. These models are imported from sklearn and keras libraries. The models used are listed below.

3.3.1 Stochastic gradient descent

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in trade for a lower convergence rate.

3.3.2 Logistic Regression

The logistic model (or logit model) is a statistical model that models the probability of an event taking place by having the log-odds for the event be a linear combination of one or more independent variables. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (the coefficients in the linear combination). Formally, in binary logistic regression there is a single binary dependent variable, coded by an indicator variable, where the two values are labeled "0" and "1", while the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name.

3.3.3 Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

3.3.4 Random Forest Algorithm

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. It is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions. They can handle skewed or categorical data (ordinal or non-ordinal). It is generally considered a very accurate predictive model.

3.3.5 XGBoost

In this algorithm, decision trees are created in sequential form. Weights play an important role in XGBoost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and these variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems**.**

3.3.6 Sequential Model

The Sequential model tends to be one of the simplest models as it constitutes a linear set of layers, whereas the functional API model leads to the creation of an arbitrary network structure. The sequential API allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs. Alternatively, the functional API allows you to create models that have a lot more flexibility as you can easily define models where layers connect to more than just the previous and next layers. In fact, you can connect layers to (literally) any other layer. As a result, creating complex networks such as siamese networks and residual networks become possible.

3.4 Modules

Python:

The main reason for using python in this project is its simplicity to learn. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Moreover, Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Since there is no compilation step, the edit test debug cycle is incredibly fast. Debugging Python programs is easy and a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. These are the various reasons for using python 3.5 in this Proposed System.

Google Colab notebook:

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook. You can download any Colab notebook that you've created from Google Drive, or from within Colab's File menu. All Colab notebooks are stored in the open source Jupyter notebook format (.ipynb). Colab notebooks are stored in Google Drive, or can be loaded from

GitHub. Colab notebooks can be shared just as you would with Google Docs or Sheets. Simply click the Share button at the top right of any Colab notebook.

Numpy:

Numpy is a python library used for working with arrays and perform some basic computation. In this system, numpy is mostly used with random function to input seed value to generate repeated random number. In some cases it is also used for conversion of int datatype into float and vice-vera.

Pandas:

Pandas is used for data analysis in this project and it provides numerous tools to analyze the data. The package comes with several data structures that can be used for many different data manipulation tasks. Pandas contain varieties of utilities to perform Input / Output operations in a seamless manner. Pandas can also read data from various forms of formats such as CSV, TSV and MS Excel etc. These are the uses of pandas in this project.

Sklearn:

The models Decision Tree, Logistic Regression, Random Forest, Stochastic Gradient Descent, XGBoost and sequential model were used in this project. To use these models, Scikit-learn (Sklearn) is required. This the most useful and robust library for machine learning in Python.

Matplotlib:

To compare accuracy of each model visually, graphs are needed. To use these bar graphs, Matplotlib is used where it is a low level graph plotting library in python serves visualization ability. Matplotlib can be used in Python scripts, the Python and IPython shell, web application servers, and various graphical user interface toolkits. Matplotlib.pyplot is acollection of command style functions that make matplotlib work like MATLAB.

Keras:

Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK. So, as Sequential model is used in this proposed model and it produces better performance measures than other models, keras is used to import this model. It cannot handle low-level computations, so it makes use of the Backend library to resolve it.

3.5 UML diagram

The Unified Modelling Language (UML) is a standard language for writing software blueprints. The UML is a language for:

- **Specifying:** It is just like a blueprint created by an architect prior to the construction.
- **Visualizing:** Visualizing is concerned with deep analysis of systems to be constructed.
- **Constructing:** Modelling also provides us with mechanisms which are essential while constructing a system.
- **Documenting:** Finally, modelling justifies its importance by applying all its credentials to be bound in a piece of paper referred to as a document

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

3.5.1 Use case diagram of Classification of garments using fashion MNIST dataset

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed together its functionalities use cases are prepared and actors are identified.

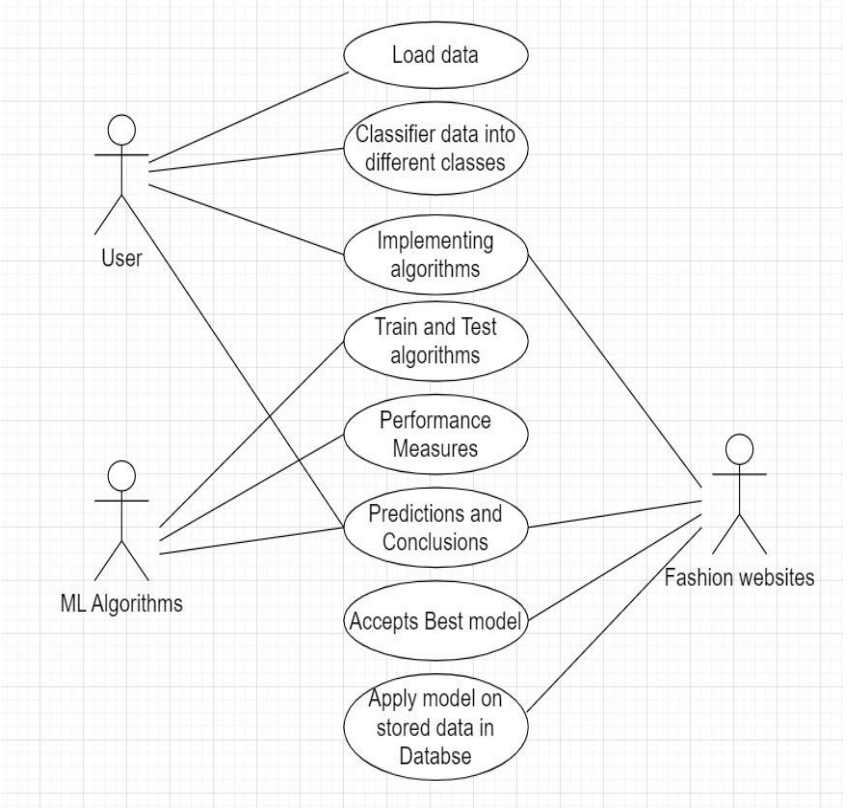

Fig. 3.5.1 Use Case Diagram of Classification of garments using fashion MNIST dataset

In Figure 3.5.1, the user loads data and classifies the data into different classes. Then different algorithms are used to train the model by using the data and to calculate performance measures. Fashion websites uses the model which has best performance measure i.e., accuracy and precision.

3.5.2 Sequence Diagram of Classification of garments using fashion MNIST dataset

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

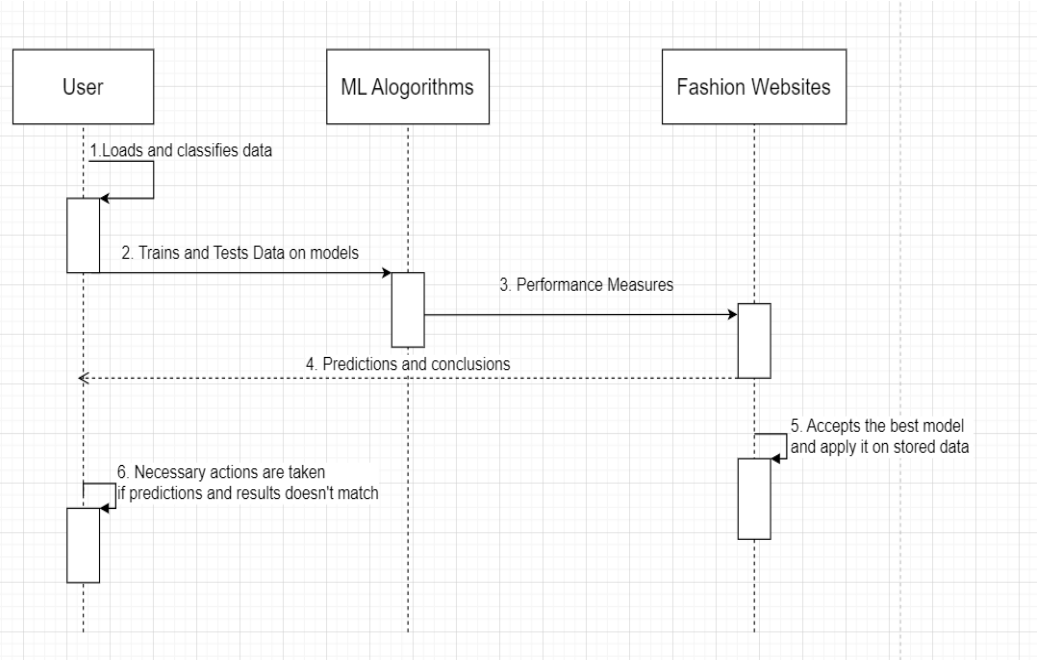

Fig. 3.5.2 Sequence Diagram of Classification of garments
using fashion MNIST dataset

The Figure 3.5.2 shows the sequence followed in the project, where the user loads data and classifies the data into different classes. Then different algorithms are used to train the model by using the data and to calculate performance measures. Fashion websites uses the model which has best performance measure i.e., accuracy and precision.

4. TESTING AND RESULTS

4.1 Testing

As we can see from the Figure 4.1.1 below the view of images based on the pixel values ranging from 0 to 255. Labels are classified into 10 different classes as T-shirt, Trouser, PullOver, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

Figure 4.1.1 Image view of different garments

Performance measures are calculated for various models used in here. Cross validation is used here to find the proper score of each model, also to ensure that the model is not overfitting or underfitting. If the cross validation score values for a performance measure (say accuracy) are not varying significantly for various folds (k-folds) then we can say that the model is not overfitting. If the cross validation score values for a performance measure (say accuracy) are not very low for various folds (k-folds) then we can say that the model is not underfitting. We will perform k-fold cross-validation. This will randomly split the training set into 3 distinct subsets called folds (cv=3). Since cross validation is a computing intensive and time consuming process, we are limiting 'cv' (no. of folds) to 3 instead of normally 10 folds. Then will train and evaluate each model 3 times by picking a different fold for evaluation every time and training on the other 2 folds. The result will be an array containing the 3 evaluation scores for each of the measures - accuracy, precision, F1 score. We will use cross_val_score() function to calculate accuracy.

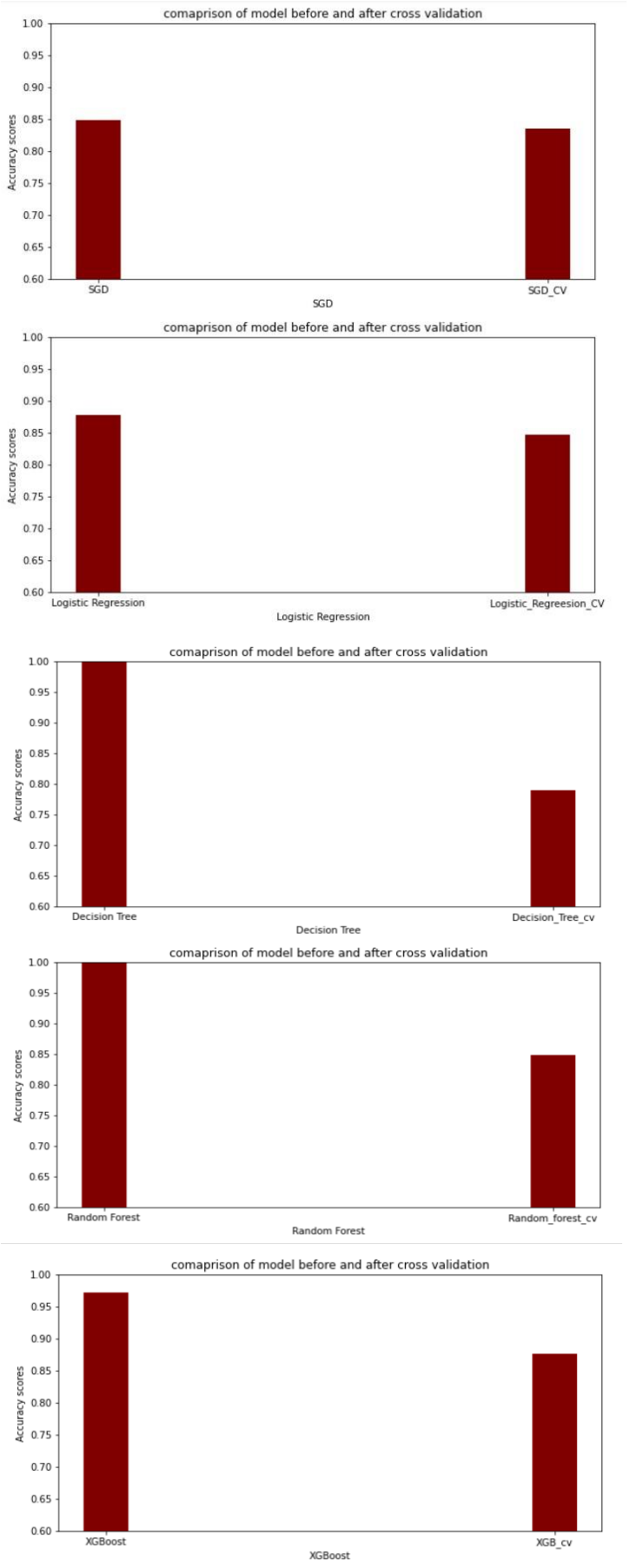

Figure 4.1.2 Comparison of accuracy of different models before and after Cross alidation

4.2 Results

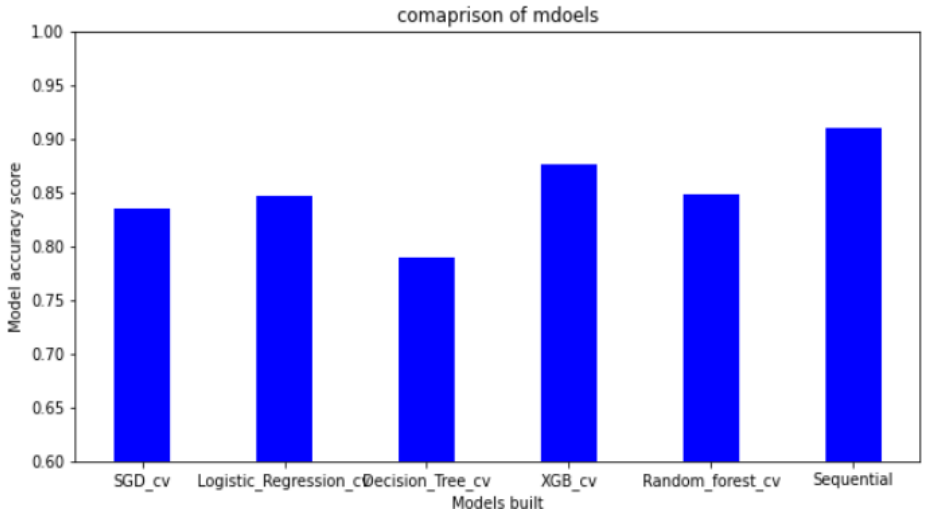

Figure 4.2.1 Comparison of all Models accuracy

From the Figure 4.2.1, the accuracy of all the models is above 80 except Decision tree, where it has around 80. XGBoost has accuracy of nearly 90 and sequential model has over 90. Because epoch operation is performed onthis model. So eventually the accuracy increases as the number of epochs increases. Hence, we can consider sequential model as best model because it has high accuracy.

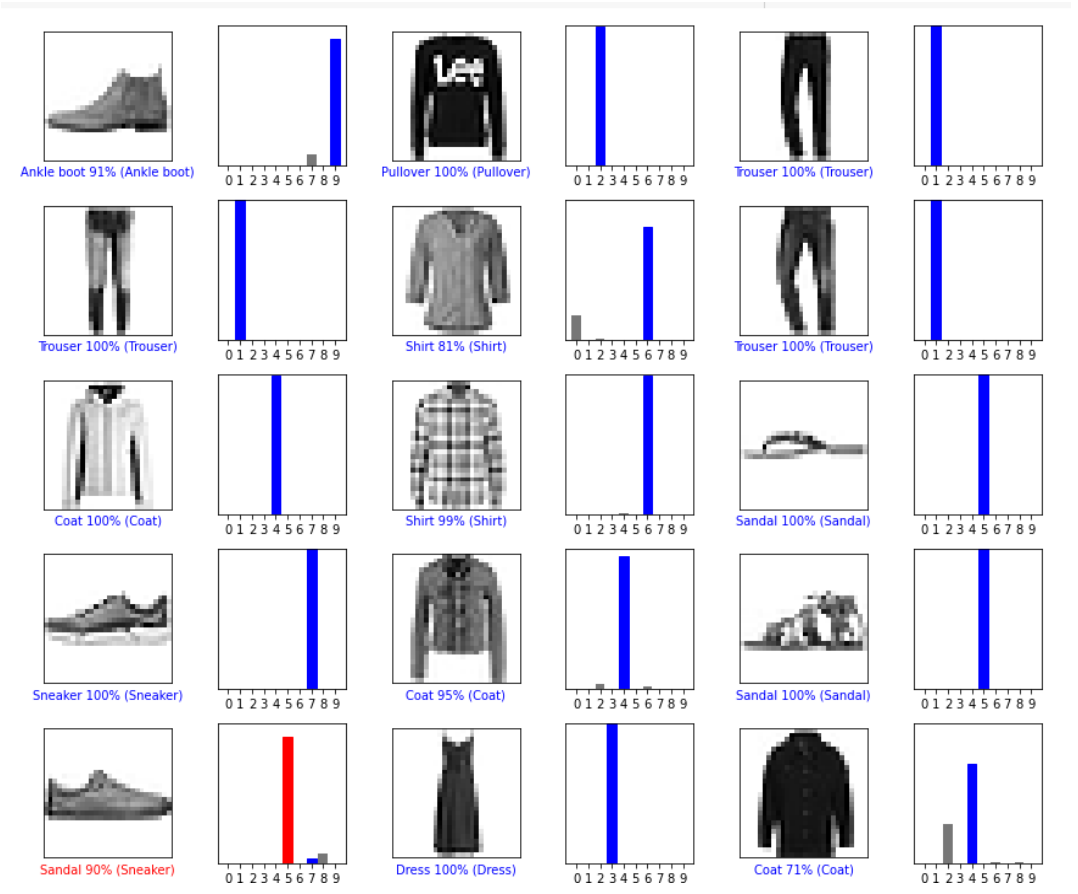

Figure 4.2.2 Results of Sequential model after testing the model

The Figure 4.2.2 describes the output for the first 15 labels and the predicted output. If the color of the bar in graph represents blue then we can say the predicted output is right and classification is done well. If the bar color is red then the classification of the model is wrong. Grey color represents some deviation from the output. If both pictures have similar structure, then there is chance of wrong prediction. As from the figure we can conclude that most of the class labels have the right outcome and classification of the model works accurately.

5. CONCLUSION

Classification has become a severe problem for many companies and websites because of the huge data and different kinds of data. In this Proposed system, garment classification is done using various models. The best model is sequential model. The final accuracy of the sequential model is 91% and is the best model when compared to all other models. This algorithm is easy to implement and fast when compared to all other models used to classify the dataset. This machine learning model can be used by fashion website to solve the classification problem. Thus, the proposed system provides accurate results and is can easily classify the data.

The similar classification system can be built to solve this problem with the help many upcoming technologies and it is suggested to further improve this model using different kind of machine learning models. Also, new algorithms can be proposed to achieve more accuracy and reliability. Some warnings had occurred while training some models. These can be eliminated by increasing the number of iterations or scale the data. The time taken for the implementation of some models and cross validation is very high. Even tough k-fold value is reduced, time consumption is high. There are some ways in which the performance of the system can be further improved and time consumption can be reduced.

BIBILOGRAPHY

[1] V.Q. Leithardt, L. Rodrigo, S. Correia, “Cloth Classifying from Fashion MNIST dataset,” 2021 Advances in Science Technology and Engineering Systems Journal, February 2021, DOI: 10.25046/aj0601109

[2] M. Kayed, A. Anter and H. Mohamed, “Classification of Garments from Using CNN LeNet-5 Architecture,” 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), Aswan, Egypt, 2020, pp. 238-243, DOI: 10.1109/ITCE48509.2020.9047776

[3] K V, Greeshma & Sreekumar, K, “Fashion-MNIST classification based on HOG feature descriptor,” 2019 International Journal of Innovative Technology and Exploring Engineering. 8. 960-962.

[4] Shen. S, “Image Classification of Fashion-MNIST Dataset Using Long Short-Term Memory Networks” 2018, Research School of Computer Science, Australian National University, Canberra

[5] A. Krizhevsky, I. Sutskever, G.E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks” 2018, Information Technology and Management Science

[6] Fashion MNIST Dataset [available online]: https://github.com/zalandoresearch/fashion-mnist

[7] Python(3.5)[Available online]: https://www.python.org/downloads/release/python-350

APPENDIX-A

```
#Classification of Garments using Fashion MNIST dataset
#Connecting to drive
from google.colab import drive
drive.mount("/content/drive")

#Importing required libraries
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import gzip
import numpy as np
import tensorflow as tf
import numpy as np

#Loading data from drive
fashion_mnist= tf.keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels)= fashion_mnist.load_data()
filePath_train_set = '/content/drive/MyDrive/data/train-images-idx3-ubyte.gz'
filePath_train_label = '/content/drive/MyDrive/data/train-labels-idx1-ubyte.gz'
filePath_test_set = '/content/drive/MyDrive/data/t10k-images-idx3-ubyte.gz'
filePath_test_label = '/content/drive/MyDrive/data/t10k-labels-idx1-ubyte.gz'

#Unzipping the data and storing it new variables
with gzip.open(filePath_train_label, 'rb') as trainLbpath:
    trainLabel = np.frombuffer(trainLbpath.read(), dtype=np.uint8,
                               offset=8)
with gzip.open(filePath_train_set, 'rb') as trainSetpath:
    trainSet = np.frombuffer(trainSetpath.read(), dtype=np.uint8,
                               offset=16).reshape(len(trainLabel), 784)
with gzip.open(filePath_test_label, 'rb') as testLbpath:
    testLabel = np.frombuffer(testLbpath.read(), dtype=np.uint8,
                               offset=8)
with gzip.open(filePath_test_set, 'rb') as testSetpath:
    testSet = np.frombuffer(testSetpath.read(), dtype=np.uint8,
                               offset=16).reshape(len(testLabel), 784)
```

```
X_train, X_test, y_train, y_test = trainSet, testSet, trainLabel, testLabel

#Giving Class labels/Names to different garments
class_names= ['T-
shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

#showImage function potrays the image and reshaping it to get 28 x 28 pixels
def showImage(data):
    some_article = data
    some_article_image = some_article.reshape(28, 28)
    plt.imshow(some_article_image, cmap = matplotlib.cm.binary, interpolation="nearest")
    plt.axis("off")
    plt.show()

showImage(X_train[0])
y_train[0]
import numpy as np
np.random.seed(42)

#We need to shuffle our training data to ensure that we don't miss out any digit in a cross validation fold.
shuffle_index = np.random.permutation(60000)
X_train, y_train = X_train[shuffle_index], y_train[shuffle_index]
showImage(X_train[0])
data={}
li=[]

#Each image (instance) in the dataset has 784 pixels (features) and value of each feature(pixel) ranges
#from 0 to 255, and this range is too wide, hence we would need to use feature scaling here to
#apply standardization to this dataset X_train, so that all the values of each feature (pixel) is in a small
#range (based on the standard deviation value).
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train.astype(np.float64))

#Importing performance measures using sklearn
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
```

```
#Stochastic Gradient Descent Classifier
from sklearn.linear_model import SGDClassifier
sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train_scaled, y_train)
y_train_predict = sgd_clf.predict(X_train[0].reshape(1, -1))
y_train_predict = sgd_clf.predict(X_train_scaled)
sgd_accuracy = accuracy_score(y_train, y_train_predict)
sgd_precision = precision_score(y_train, y_train_predict, average='weighted')
sgd_recall = recall_score(y_train, y_train_predict, average='weighted')
sgd_f1_score = f1_score(y_train, y_train_predict, average='weighted')
data['SGD']=sgd_accuracy
sgd={}
sgd['SGD']=sgd_accuracy
li.append(sgd)
#print(li)
print("SGD Accuracy: ", sgd_accuracy)
print("SGD Precision: ", sgd_precision)
print("SGD Recall: ", sgd_recall)
print("SGD F1 Score: ", sgd_f1_score)

#Logistic Regression
from sklearn.linear_model import LogisticRegression
# using Softmax Regression (multi-class classification problem)
log_clf = LogisticRegression(multi_class="multinomial", solver="lbfgs", C=10, random_state=42)
# 'C' is hyperparameter for regularizing L2
# 'lbfgs' is Byoden-Fletcher-Goldfarb-Shanno(BFGS) algorithm
log_clf.fit(X_train_scaled, y_train)

# Let us predict some instance from the dataset using the above trained model
y_train_predict = log_clf.predict(X_train[0].reshape(1, -1))
# Let us predict all instances of training dataset X_train_scaled using the above trained model
y_train_predict = log_clf.predict(X_train_scaled)
log_accuracy = accuracy_score(y_train, y_train_predict)
log_precision = precision_score(y_train, y_train_predict, average='weighted')
log_recall = recall_score(y_train, y_train_predict, average='weighted')
log_f1_score = f1_score(y_train, y_train_predict, average='weighted')
data['Logistic Regression']=log_accuracy
log={}
```

```
log['Logistic Regression']=log_accuracy
li.append(log)
print("Logistic Accuracy: ", log_accuracy)
print("Logistic Precision: ", log_precision)
print("Logistic Recall: ", log_precision)
print("Logistic F1 Score: ", log_f1_score)

#Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dec_tree_clf = DecisionTreeClassifier(max_depth=50, random_state=42)
# Scaling is not needed for Decision Tree algorithm and hence for Random Forest and XGBoost algorithm
# as they are also based on Decision Trees. Hence, not using scaled training dataset here
dec_tree_clf.fit(X_train, y_train)
# Let us predict some instance from the dataset using the above trained model
y_train_predict = dec_tree_clf.predict(X_train[0].reshape(1, -1))
# Let us predict all instances of training dataset X_train using the above trained model
y_train_predict = dec_tree_clf.predict(X_train)
dec_tree_accuracy = accuracy_score(y_train, y_train_predict)
dec_tree_precision = precision_score(y_train, y_train_predict, average='weighted')
dec_tree_recall = recall_score(y_train, y_train_predict, average='weighted')
dec_tree_f1_score = f1_score(y_train, y_train_predict, average='weighted')
data['Decision Tree']=dec_tree_accuracy
dec={}
dec['Decision Tree']=dec_tree_accuracy
li.append(dec)
print("Decision Tree Accuracy: ", dec_tree_accuracy)
print("Decision Tree Precision: ", dec_tree_precision)
print("Decision Tree Recall: ", dec_tree_precision)
print("Decision Tree F1 Score: ", dec_tree_f1_score)

#Random Forest
from sklearn.ensemble import RandomForestClassifier
rnd_clf = RandomForestClassifier(n_estimators=100, max_depth=50, random_state=42)
rnd_clf.fit(X_train, y_train)
# Let us predict some instance from the data set using the above trained model
y_train_predict = rnd_clf.predict(X_train[0].reshape(1, -1))
y_train_predict = rnd_clf.predict(X_train)
```

```
rnd_accuracy = accuracy_score(y_train, y_train_predict)
rnd_precision = precision_score(y_train, y_train_predict, average='weighted')
rnd_recall = recall_score(y_train, y_train_predict, average='weighted')
rnd_f1_score = f1_score(y_train, y_train_predict, average='weighted')
data['Random Forest']=rnd_accuracy
rnd={}
rnd['Random Forest']=rnd_accuracy
li.append(rnd)
print("Random Forest Accuracy: ", rnd_accuracy)
print("Random Forest Precision: ", rnd_precision)
print("Random Forest Recall: ", rnd_precision)
print("Random Forest F1 Score: ", rnd_f1_score)

#XGboost
from xgboost import XGBClassifier
xgb_clf = XGBClassifier(n_estimators=20, max_depth=10, random_state=42)
xgb_clf.fit(X_train, y_train)
# Let us predict some instance from the data set using the above trained model
y_train_predict = xgb_clf.predict(X_train[0].reshape(1, -1))
# Let us predict all instances of training dataset X_train using the above trained model
y_train_predict = xgb_clf.predict(X_train)
xgb_accuracy = accuracy_score(y_train, y_train_predict)
xgb_precision = precision_score(y_train, y_train_predict, average='weighted')
xgb_recall = recall_score(y_train, y_train_predict, average='weighted')
xgb_f1_score = f1_score(y_train, y_train_predict, average='weighted')
data['XGBoost']=xgb_accuracy
xgb={}
xgb['XGBoost']=xgb_accuracy
li.append(xgb)
print("XGBoost Accuracy: ", xgb_accuracy)
print("XGBoost Precision: ", xgb_precision)
print("XGBoost Recall: ", xgb_precision)
print("XGBoost F1 Score: ", xgb_f1_score)
Models = list(data.keys())
acc_values = list(data.values())

fig = plt.figure(figsize = (10, 5))
```

```
# Creating the bar plot to compare all models visually
plt.bar(Models, acc_values, color ='maroon',
        width = 0.4)
plt.xlabel("Models built")
plt.ylabel("Model accuracy score")
plt.title("comaprison of mdoels")
plt.show()

# Function to calculate mean and standard deviation of each score (e.g. accuracy, precision, etc.)
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

#Using cross validation to find the proper score of each model, also to ensure that the model is
#not overfitting or underfitting.
# If the cross validation score values for a performance measure (say accuracy) are not varying
#significantly for various folds (k-folds) then we can say that the model is not overfitting.
#If the cross validation score values for a performance measure (say accuracy) are not very low for various
#folds (k-folds) then we can say that the model is not underfitting.
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
data_cv={}

#Stochastic Gradient Descent Classifier
sgd_scores = cross_val_score(sgd_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
display_scores(sgd_scores)
sgd_accuracy_cv = sgd_scores.mean()
y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
confusion_matrix(y_train, y_train_pred)
```

```
sgd_precision_cv = precision_score(y_train, y_train_pred, average='weighted')
sgd_recall_cv = recall_score(y_train, y_train_pred, average='weighted')
sgd_f1_score_cv = f1_score(y_train, y_train_pred, average='weighted')

print("SGD CV Accuracy: ", sgd_accuracy_cv)
print("SGD CV Precision: ", sgd_precision_cv)
print("SGD CV Recall: ", sgd_precision_cv)
print("SGD CV F1 Score: ", sgd_f1_score_cv)
data_cv['SGD_cv']=sgd_accuracy_cv
li[0]['SGD_CV']=sgd_accuracy_cv
print(li)

#Logistic Regression
log_scores = cross_val_score(log_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
display_scores(log_scores)
log_accuracy_cv = log_scores.mean()

y_train_pred = cross_val_predict(log_clf, X_train_scaled, y_train, cv=3)
confusion_matrix(y_train, y_train_pred)
log_precision_cv = precision_score(y_train, y_train_pred, average='weighted')
log_recall_cv = recall_score(y_train, y_train_pred, average='weighted')
log_f1_score_cv = f1_score(y_train, y_train_pred, average='weighted')

print("Logistic CV Accuracy: ", log_accuracy_cv)
print("Logistic CV Precision: ", log_precision_cv)
print("Logistic CV Recall: ", log_precision_cv)
print("Logistic CV F1 Score: ", log_f1_score_cv)
data_cv['Logistic_Regression_cv']=log_accuracy_cv
li[1]['Logistic_Regreesion_CV']=log_accuracy_cv
print(li)

# Scaled Features not required for Decision Tree
dec_tree_scores = cross_val_score(dec_tree_clf, X_train, y_train, cv=3, scoring="accuracy")
display_scores(dec_tree_scores)
dec_tree_accuracy_cv = dec_tree_scores.mean()

y_train_pred = cross_val_predict(dec_tree_clf, X_train, y_train, cv=3)
```

```
confusion_matrix(y_train, y_train_pred)

dec_tree_precision_cv = precision_score(y_train, y_train_pred, average='weighted')
dec_tree_recall_cv = recall_score(y_train, y_train_pred, average='weighted')
dec_tree_f1_score_cv = f1_score(y_train, y_train_pred, average='weighted')

print("Decision Tree CV Accuracy: ", dec_tree_accuracy_cv)
print("Decision Tree CV Precision: ", dec_tree_precision_cv)
print("Decision Tree CV Recall: ", dec_tree_precision_cv)
print("Decision Tree CV F1 Score: ", dec_tree_f1_score_cv)


data_cv['Decision_Tree_cv']=dec_tree_accuracy_cv
li[2]['Decision_Tree_cv']=dec_tree_accuracy_cv
print(li)

# Scaled features not required for XGBoost (as it is based on Decision Trees)
xgb_scores = cross_val_score(xgb_clf, X_train, y_train, cv=3, scoring="accuracy")
display_scores(xgb_scores)
xgb_accuracy_cv = xgb_scores.mean()

y_train_pred = cross_val_predict(xgb_clf, X_train, y_train, cv=3)
confusion_matrix(y_train, y_train_pred)
xgb_precision_cv = precision_score(y_train, y_train_pred, average='weighted')
xgb_recall_cv = recall_score(y_train, y_train_pred, average='weighted')
xgb_f1_score_cv = f1_score(y_train, y_train_pred, average='weighted')

print("XGBoost CV Accuracy: ", xgb_accuracy_cv)
print("XGBoost CV Precision: ", xgb_precision_cv)
print("XGBoost CV Recall: ", xgb_precision_cv)
print("XGBoost CV F1 Score: ", xgb_f1_score_cv)
data_cv['XGB_cv']=xgb_accuracy_cv
li[4]['XGB_cv']=xgb_accuracy_cv
print(li)

# Scaled features not required for Random Forest Classifier
rnd_clf = RandomForestClassifier(n_estimators=20, max_depth=10, random_state=42)
rnd_clf.fit(X_train, y_train)
rnd_scores = cross_val_score(rnd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

```
display_scores(rnd_scores)
rnd_accuracy_cv = rnd_scores.mean()

y_train_pred = cross_val_predict(rnd_clf, X_train, y_train, cv=3)
confusion_matrix(y_train, y_train_pred)
rnd_precision_cv = precision_score(y_train, y_train_pred, average='weighted')
rnd_recall_cv = recall_score(y_train, y_train_pred, average='weighted')
rnd_f1_score_cv = f1_score(y_train, y_train_pred, average='weighted')

print("Random Forest CV Accuracy: ", rnd_accuracy_cv)
print("Random Forest CV Precision: ", rnd_precision_cv)
print("Random Forest CV Recall: ", rnd_precision_cv)
print("Random Forest CV F1 Score: ", rnd_f1_score_cv)
data_cv['Random_forest_cv']=rnd_accuracy_cv
li[3]['Random_forest_cv']=rnd_accuracy_cv
print(li)
for i in li:
  Model = list(i.keys())
  acc_values = list(i.values())

  fig = plt.figure(figsize = (10, 5))

  #Creating the bar plot to compare models accuracy before and after cross validation
  plt.bar(Model, acc_values, color ='maroon',width = 0.1)
  plt.ylim(0.6,1.0)
  plt.xlabel(Model[0])
  plt.ylabel("Accuracy scores")
  plt.title("comaprison of model before and after cross validation")
  plt.show()
train_images= train_images/255.0
test_images= test_images/255.0
model= tf.keras.Sequential([
  tf.keras.layers.Flatten(input_shape=(28,28)),
  tf.keras.layers.Dense(128,  activation='relu'),
  tf.keras.layers.Dense(10)
])
model.compile( optimizer='adam', loss= tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
            metrics=['accuracy'])
```

```
model.fit(train_images, train_labels, epochs=10)
y = np.array([0.82,0.86,0.87,0.88,0.89,0.896,0.898,0.9,0.907,0.91])
x = np.array([1,2,3,4,5,6,7,8,9,10])

plt.plot(x, y) # Plot the chart
plt.xlabel("epoch") # add X-axis label
plt.ylabel("accuracy") # add Y-axis label
plt.title("Accuracy")  # add title
plt.show()

data_cv['Sequential']=y[-1]
test_loss, test_accu= model.evaluate(test_images, test_labels, verbose=2)
print('\n Test Accuracy', test_accu)
Models = list(data_cv.keys())
acc_values = list(data_cv.values())
fig = plt.figure(figsize = (10, 5))
plt.bar(Models, acc_values, color ='blue',
        width = 0.4)
plt.ylim(0.6,1.0)
plt.xlabel("Models built")
plt.ylabel("Model accuracy score")
plt.title("comaprison of mdoels")
plt.show()

#Sequential model
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
predictions[0]
np.argmax(predictions[0])
test_labels[0]

def plot_image(i, predictions_array, true_label, img):
  true_label, img = true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
```

```
  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  true_label = true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')

# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(num_rows, 2*num_cols, 2*i+2)
plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```

APPENDIX-B

This section describes the user manual under which it compiles two sections. They are software and technology used in our model and the project's execution steps.

User Manual

The technology and software requirements are listed. There is no need for installation of any software as the google colab gives us direct access through the internet connection.

- **Software Requirements**:
  - The system needs an operating system of windows type and the language used for the program execution is python.
- **Technology**:
  - Google Colab:
    - Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.
    - With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine.
  - Python:
    - Python is an interpreted high-level programming language for general purpose programming. Created by Guido van Rossum, Python has adesign philosophy that emphasizes code readability, notably using significant whitespace.
    - It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management.
    - Python interpreters are available for many operating systems.

Code Implementation

1. We have imported the libraries like pandas, numpy, sklearn, keras, matplotlib.
2. We have the dataset which is uploaded into drive from zolando website.
3. Standardizing the dataset, to make sure that there is no bias.
4. For every model, we are using sklearn to import that model and training the model using training dataset.
5. Cross validation is applied on each model to find out whether the model is overfitting or underfitting
6. Performance measures are calculated for each model.
7. Comparing each model's performance measures, before and after cross validation.
8. The model which has best performance measures is selected.
9. The final model is tested using testing dataset.