# Image Classification And Logistic Regression

**A MINI-PROJECT REPORT**

***Submitted By:***

**DEVANSH AWASTHI- ENG17CS0065**

**DEEPAK PUROHIT- ENG17CS0062**

**BVM ANIRUDH- ENG17CS0048**

**DHRUVA SANTOSH- ENG17CS0070**

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

***at***

**DAYANANDA SAGAR UNIVERSITY**

**SCHOOL OF ENGINEERING, BANGALORE-560068**

**7TH SEMESTER**

**(Course Code:16CS401)**

**MACHINE LEARNING LABORATORY**

**DAYANANDA SAGAR UNIVERSITY**

**CERTIFICATE**

This is to certify that the Machine Learning Mini-Project report entitled **“Image Classification and Logistic Regression”** being submitted by Devansh Awasthi (ENG17CS0065), Deepak Purohit (ENG17CS0062), BVM Anirudh (ENG17CS0048) and Dhruva Santosh (ENG17CS0070) to Department of Computer Science and Engineering, School of Engineering, Dayananda Sagar University, Bangalore, for the 7th semester B.Tech C.S.E of this university during the academic year 2021.

***Date***:___________

____________________________

***Signature of the Faculty in Charge***

____________________________

***Signature of the Chairman***

## ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We are especially thankful to our **Chairman ,Dr. Sanjay Chitnis**, for providing necessary departmental facilities, moral support and encouragement.

We are very much thankful to **Dr. S.G Shaila, Associate Professor**, for providing help and suggestions in completion of this mini project successfully.

We have received a great deal of guidance and co-operation from our friends and we wish to thank all that have directly or indirectly helped us in the successful completion of this project work.

Devansh Awasthi- (ENG17CS0065)
Deepak Purohit- (ENG17CS0062)
BVM Anirudh- (ENG17CS0048)
Dhruva Santosh- (ENG17CS0070)

# TABLE OF CONTENTS

| SL.NO | TITLE | PAGE. NO |
|---|---|---|
| i. | Cover Page | 1 |
| ii. | Certificate | 2 |
| iii. | Acknowledgement | 3 |
| iv. | Content | 4 |
| v | Abstract | 5 |
| 1. | Introduction | 6 |
| 2. | Problem Statement | 7 |
| 3. | Literature Survey | 8 |
| 4. | Methodology | 9 |
| 5. | Requirement Analysis | 5 |
| 6. | Design | 14 |
| 7. | Code | 18 |
| 8. | Results / Screenshots | 23 |
| 9. | Conclusion | 27 |
| 10. | References | 28 |

# ABSTRACT

Our main aim is to train a neural network model to classify images of clothing and apply Linear Analysis to predict the movement of the market.

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general-purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer: whereas in previous “parameter server” designs the management of shared state is built into the system, TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks. Several Google services use TensorFlow in production, we have released it as an open-source project, and it has become widely used for machine learning research. In this paper, we describe the TensorFlow dataflow model and demonstrate the compelling performance that Tensor- Flow achieves for several real-world applications.

Logistic regression is one of the most common multivariate analysis models utilized in expidemiology. It allows the measurement of the association between the occurrence of an event (qualitative dependent variable) and factors susceptible to influence it (explicative variables). The choice of explicative variables that should be included in the logistic regression model is based on prior knowledge of the disease physiopathology and the statistical association between the variable and the event, as measured by the odds ratio. The main steps for the procedure, the conditions of application, and the essential tools for its interpretation are discussed concisely. We also discuss the importance of the choice of variables that must be included and retained in the regression model in order to avoid the omission of important confounding factors. Finally, by way of illustration, we provide an example from the literature, which should help the reader test his or her knowledge.

# 1. INTRODUCTION

The clothing classification problem is a new standard dataset used in computer vision and deep learning. Although the dataset is relatively simple, it can be used as the basis for learning and practicing how to develop, evaluate, and use deep convolutional neural networks for image classification from scratch. This includes how to develop a robust test harness for estimating the performance of the model, how to explore improvements to the model, and how to save the model and later load it to make predictions on new data. Fashion MNIST dataset which contains 70,000 grayscale images in 10 categories. The images show individual articles os clothing at low resolution (28 by 28 pixels).The images are 28x28 NumPy arrays, with pixel values ranging from 0 to 255. The labels are an array of integers, ranging from 0 to 9.

As the name already indicates, logistic regression is a regression analysis technique. Regression analysis is a set of statistical processes that you can use to estimate the relationships among variables. More specifically, you use this set of techniques to model and analyze the relationship between a dependent variable and one or more independent variables. Regression analysis helps you to understand how the typical value of the dependent variable changes when one of the independent variables is adjusted and others are held fixed.

## 2. PROBLEM STATEMENT

The goal of this project is-

- To classify images of clothing: To train a neural network model to classify images of clothing, like sneakers and shirts using uses tf. Keras, a high-level API to build and train models in TensorFlow.

- To apply Linear Discriminant Analysis to predict the movement of the market for the given Smarket data set consisting of percentage returns for the S&P 500 stock index over 1,250 days before 2005.

## 3. LITERATURE SURVEY

"Classification of fashion article images using convolutional neural networks" author Shobhit Bhatnagar, Deepanway Ghosal, Maheshkumar H. Kolekar in 2017, In this paper, we propose a state-of-the-art model for classification of fashion article images. We trained convolutional neural network based deep learning architectures to classify images in the Fashion-MNIST dataset. We have proposed three different convolutional neural network architectures and used batch normalization and residual skip connections for ease and acceleration of the learning process. Our model shows impressive results on the benchmark dataset of Fashion-MNIST. Comparisons show that our proposed model reports improved accuracy of around 2% over the existing state-of-the-art systems in literature.

"A New Approach of Stock Price Prediction Based on Logistic Regression Model" author Jibing Gong; Shengtao Sun in 2009, In this paper, we present a new approach based on Logistic Regression to predict stock price trend of next month according to current month. Characteristics of our method include: (1) Feature Index Variables are easy to both understand for the private investor and obtain from daily stock trading information. (2) the prediction procedure includes a unique and crucial operation of selecting optimizing prediction parameters. (3) significant time-effectiveness and strong purposefulness enable users to predict stock price trend of next month just through considering current monthly financial data instead of needing a long term procedure of analyzing and collecting financial data. Shenzhen Development stock A (SDSA) from RESSET Financial Research Database is chosen as a study case. The SDSApsilas daily integrated data of three years from 2005 to 2007 is used to train and test our model. Our experiments show that prediction accuracies reach as high as at least 83%. In contrast to other methods, e.g. RBF-ANN prediction model, our model is lower in complexity and better accuracy in prediction

# 4. METHODOLOGY

**Tensorflow:**

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

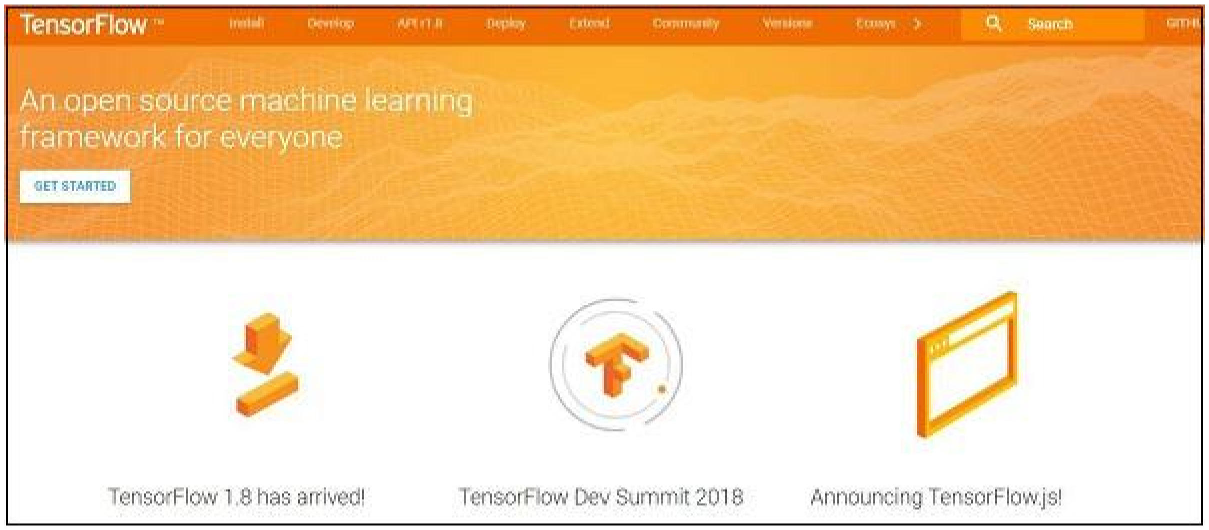

TensorFlow™
Install
Develop
API r1.8
Deploy
Extend
Community
Versions
Ecosys
Search
An open source machine learning framework for everyone
GET STARTED
TensorFlow 1.8 has arrived!
TensorFlow Dev Summit 2018
Announcing TensorFlow.js!

Fig: 4.1: Tensorflow

Let us now consider the following important features of TensorFlow −

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

## Keras:

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features −

- Consistent, simple and extensible API.
- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

## Logistic Regression:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

### Types of Logistic Regression

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types −

**(i) Binary or Binomial**

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

**(ii) Multinomial**

In such a kind of classification, dependent variable can have 3 or more possible **unordered** types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

**(iii) Ordinal**

In such a kind of classification, dependent variable can have 3 or more possible **ordered** types or the types having a quantitative significance. For example, these variables may represent “poor” or “good”, “very good”, “Excellent” and each category can have the scores like 0,1,2,3.

**Assumptions**

Before diving into the implementation of logistic regression, we must be aware of the following assumptions about the same -

- In case of binary logistic regression, the target variables must be binary always and the desired outcome is represented by the factor level 1.
- There should not be any multi-collinearity in the model, which means the independent variables must be independent of each other.
- We must include meaningful variables in our model.
- We should choose a large sample size for logistic regression.

# 5. REQUIREMENT ANALYSIS

## 5.1 <u>Software Requirements:</u>

- Anaconda environment
- Jupyter Notebook
- R Studio
- Python3 with libraries such as TensorFlow, Keras, pandas etc.
- Window 8 or above
- Pip19.0 or above

## 5.2 <u>Hardware Requirements:</u>

- Processor: i5 or above
- RAM: 8 GB
- Hard Disk: 1 TB

# 6. DESIGN

## 6.1 TENSORFLOW KERAS BASIC CLASSIFICATION MODEL

**Import the Fashion MNIST dataset**

We use the Fashion MNIST dataset which contains 70,000 grayscale images in 10 categories. The images show individual articles os clothing at low resolution (28 by 28 pixels)

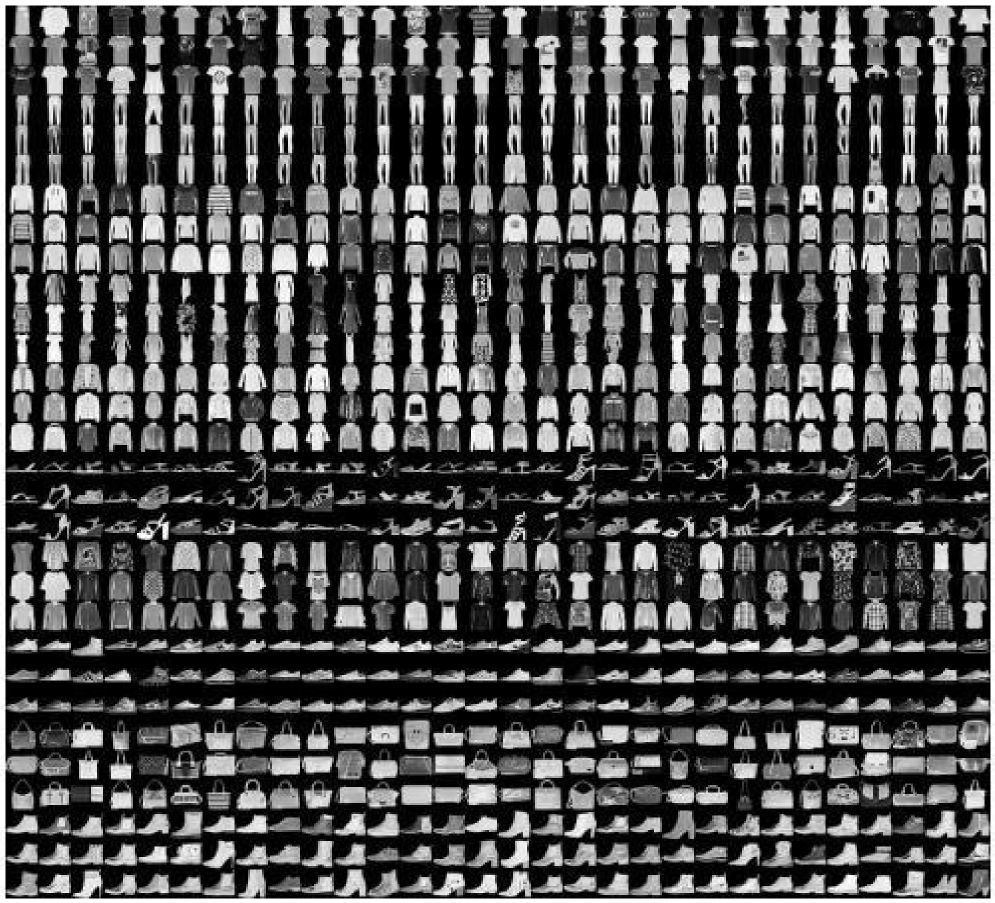
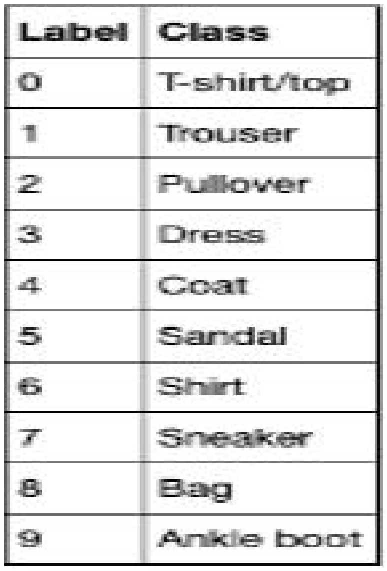


Fig 6.1.1: MNIST dataset

The images are 28x28 NumPy arrays, with pixel values ranging from 0 to 255. The labels are an array of integers, ranging from 0 to 9. These correspond to the class of clothing the image represents:



| Label | Class |
|---|---|
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

Fig 6.1.2: MNIST classes

**Explore the data**

Let's explore the format ds the dataset before training the model. The following shows there are 60,000 images in the training set, with each image represented as 28 x 28 pixels.

**Preprocess the data**

The data must be preprocessed before training the network. Is you inspect the first image in the training set, you will see that the pixel values fall in the range of 0 to 255

**Build the model**

Building the neural network requires configuring the layers of the model, then compiling the model.

**Train the model**

Training the neural network model requires the following steps:

- Feed the training data to the model. In this example, the training data train_images and
- The model learns to associate images and labels.
- You ask the model to make predictions about a test set test_images array.
- Verify that the predictions match the labels from the

**Feed the model**

To start training, call the model. sit method—so-called because it "sits" the model to the training data.

**Make predictions**

With the model trained, you can use it to make predictions about some images. The model's linear outputs, logits. Attach a softmax layer to convert the logits to probabilities, which are easier to interpret.

**Verify predictions**

With the model trained, you can use it to make predictions about some images. Correct prediction labels are blue and incorrect prediction labels are red. The number gives the percentage (out of 100) for the predicted label.

**Use the trained model**

Finally, use the trained model to predict a single image.

## 6.2 LDA PREDICTION

**Import & Explore the Smarket Dataset**

The data set consists of percentage returns for the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005. For each date, the percentage returns for each of the five previous trading days, Lag1 through Lag5, have been recorded. Also recorded are Volume (the number of shares traded on the previous day in billions), Today(the percentage return on the date in question) and Direction (whether the market was Up or Down on this date).

**Pre-process the data**

The data must be preprocessed before training the network. Split the data into training and testing data sets so that we can assess how well our model performs on an out-of-sample data set.

**To train the model**

We are using 2001 to 2004 Lag1 & Lag 2 data & the Direction. To test the model we are using 2005 data. Train the model In Python, we can sit a LDA model using Linear Discriminant Analysis function which is a part of the discriminant analysis module of the sklearn library.

**Feed the model**

To start training, call the model.fit method—so-called because it "sits" the model to the training data.

**Make predictions**

With the model trained, we can use it to make predictions. The predict() function can be used to predict the probability that the market will go up given values of the predictors.

## 7. CODE

**7.1 To classify images of clothing: To train a neural network model to classify images of clothing, like sneakers and shirts using uses tf. Keras, a high-level API to build and train models in TensorFlow.**

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag',
               'Ankle boot']

  train_images.shape

  len(train_labels)

  train_labels

  test_images.shape

  len(test_labels)

  plt.figure()
  plt.imshow(train_images[0])
  plt.colorbar()
  plt.grid(False)
  plt.show()
```

```
train_images = train_images / 255.0
test_images = test_images / 255.0

plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
model = keras.Sequential([
keras.layers.Flatten(input_shape=(28, 28)),
keras.layers.Dense(128, activation='relu'),
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)

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)

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')
 i = 0
 plt.figure(figsize=(6,3))
 plt.subplot(1,2,1)
 plot_image(i, predictions[i], test_labels, test_images)
 plt.subplot(1,2,2)
 plot_value_array(i, predictions[i],  test_labels)
 plt.show()
```

```
  i = 12
  plt.figure(figsize=(6,3))
  plt.subplot(1,2,1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(1,2,2)
  plot_value_array(i, predictions[i],  test_labels)
  plt.show()

  # 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()

 # Grab an image from the test dataset.
 img = test_images[1]
 print(img.shape)
 # Add the image to a batch where it's the only member.
 img = (np.expand_dims(img,0))
 print(img.shape)

 predictions_single = probability_model.predict(img)
 print(predictions_single)
 plot_value_array(1, predictions_single[0], test_labels)
 _ = plt.xticks(range(10), class_names, rotation=45)
np.argmax(predictions_single[0])
```

## 7.2 Linear Discriminant Analysis to predict the movement of the market for the given Smarket data set consisting of percentage returns for the S&P 500 stock index over 1,250 days before 2005.

```
library (ISLR)
names(Smarket )
dim(Smarket )
summary (Smarket )
head(Smarket)
pairs(Smarket) # Scatter Plot matrix (n*n , n-any variable)
cor(Smarket[,-9] )
attach(Smarket)
plot(Volume)
glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume , data=Smarket ,family
=binomial )
summary(glm.fits)
summary(glm.fits)$coef
glm.probs=predict (glm.fits,type ="response")
glm.probs [1:10] # vector contains 1250 obs with some probability value that it will go
Up.contrasts(Direction) glm.pred=rep("Down" ,1250)
glm.pred[glm.probs >.5]=" Up"
table(glm.pred ,Direction)
mean(glm.pred== Direction)
train =(Year <2005)
Smarket.2005= Smarket [! train ,]
dim(Smarket.2005)
Direction.2005= Direction [! train]
glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
            data=Smarket ,family =binomial ,subset =train )
glm.probs =predict (glm.fits,Smarket.2005 , type="response")
glm.pred=rep ("Down" ,252)
glm.pred[glm.probs >.5]="Up"
table(glm.pred ,Direction.2005)
mean(glm.pred== Direction.2005)
mean(glm.pred!= Direction.2005)
```

## 2. OUTPUT SCREENSHOT

### 8.1 TENSORFLOW KERAS BASIC CLASSIFICATION MODEL

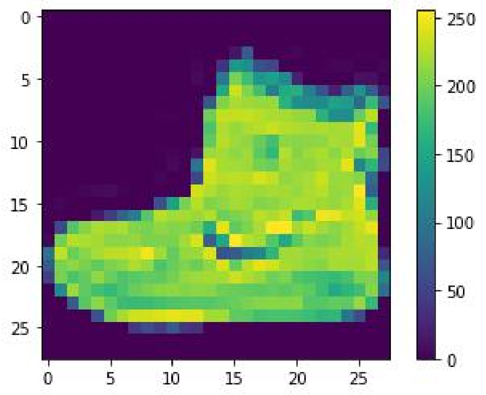
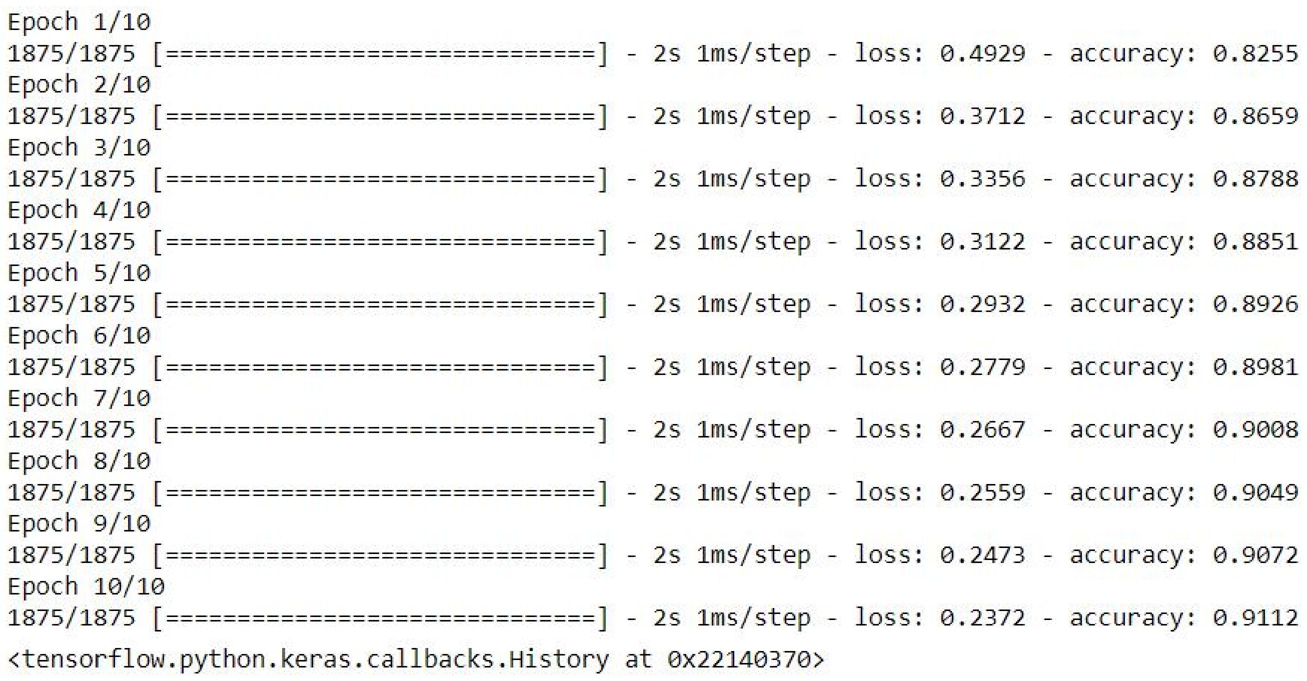


Figure 8.1.1: Ankle Boot



```
Epoch 1/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.4929 - accuracy: 0.8255
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3712 - accuracy: 0.8659
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3356 - accuracy: 0.8788
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3122 - accuracy: 0.8851
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2932 - accuracy: 0.8926
Epoch 6/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2779 - accuracy: 0.8981
Epoch 7/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2667 - accuracy: 0.9008
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2559 - accuracy: 0.9049
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2473 - accuracy: 0.9072
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2372 - accuracy: 0.9112
<tensorflow.python.keras.callbacks.History at 0x22140370>
```

Figure 8.1.2: Loss And Accuracy Metrics

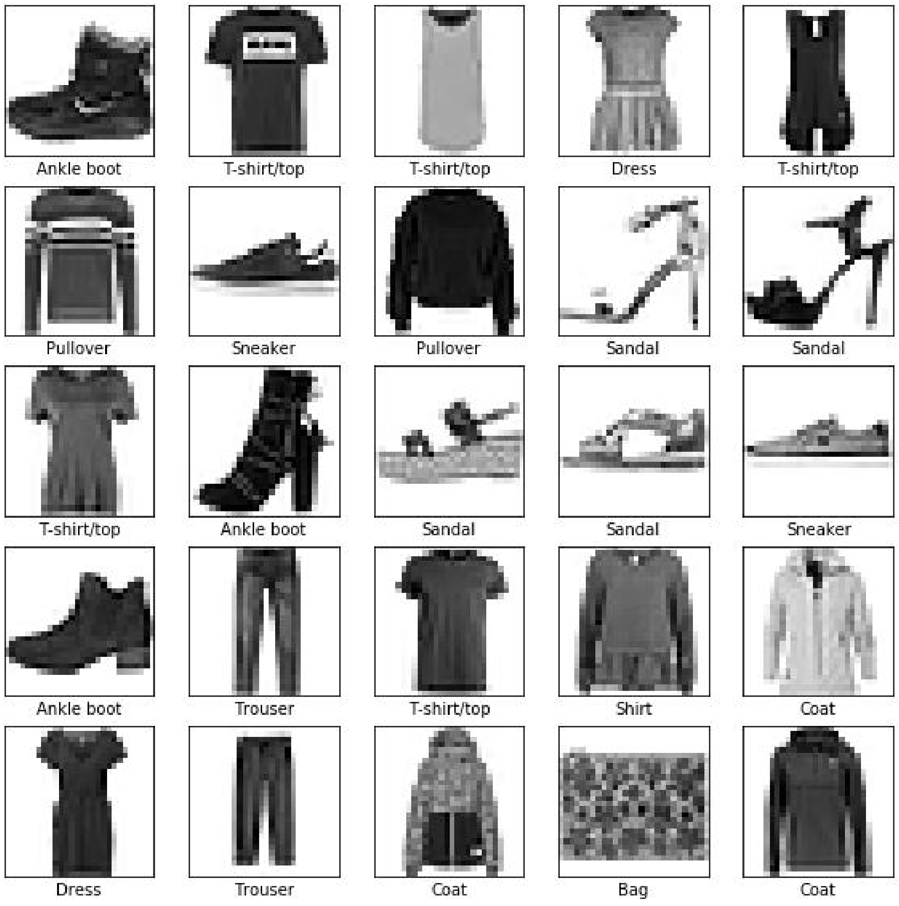


Figure 8.1.3: Testing data Set

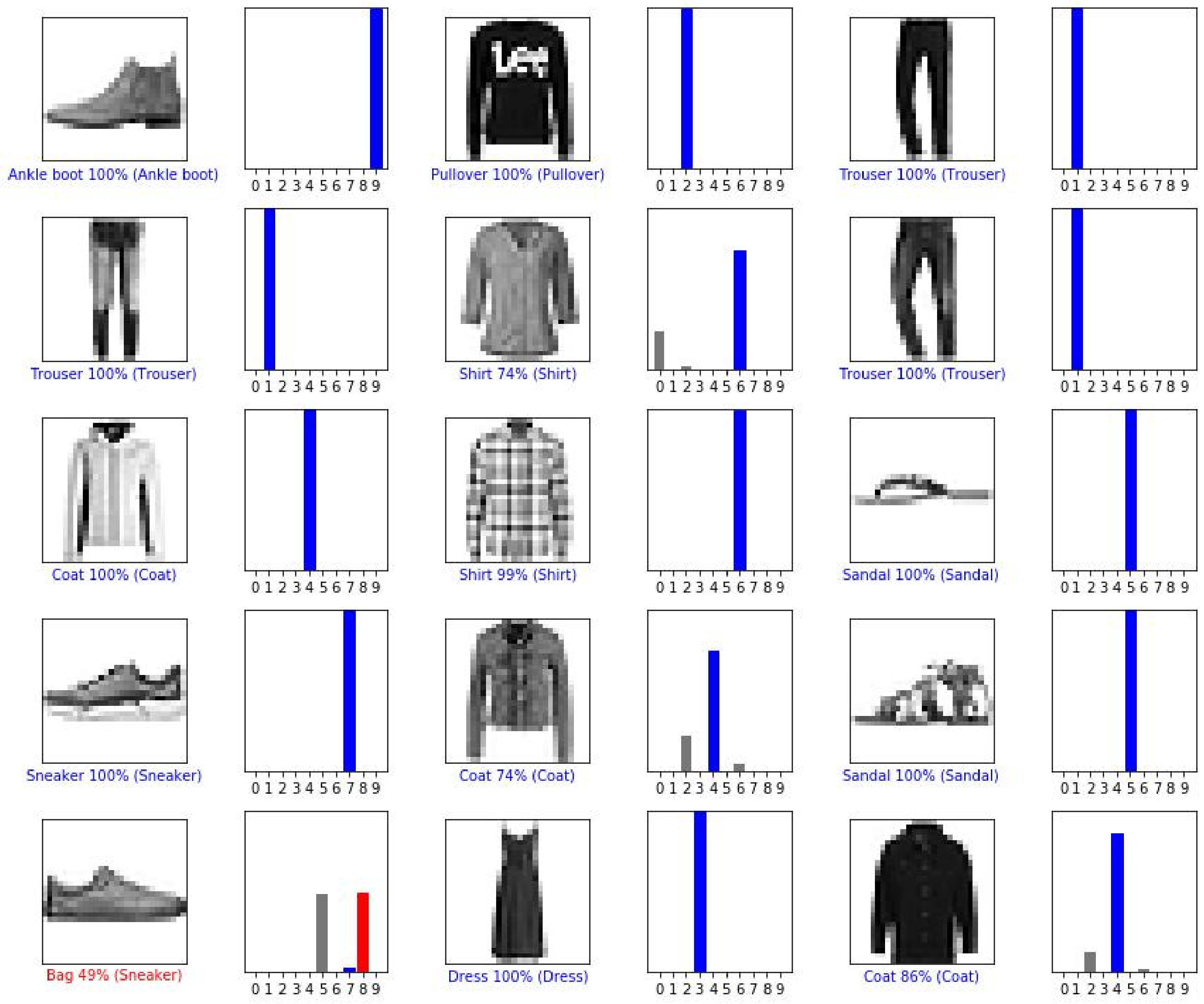


Figure 8.1.4: Training Data Set

## 8.2 Logistic Regression Prediction

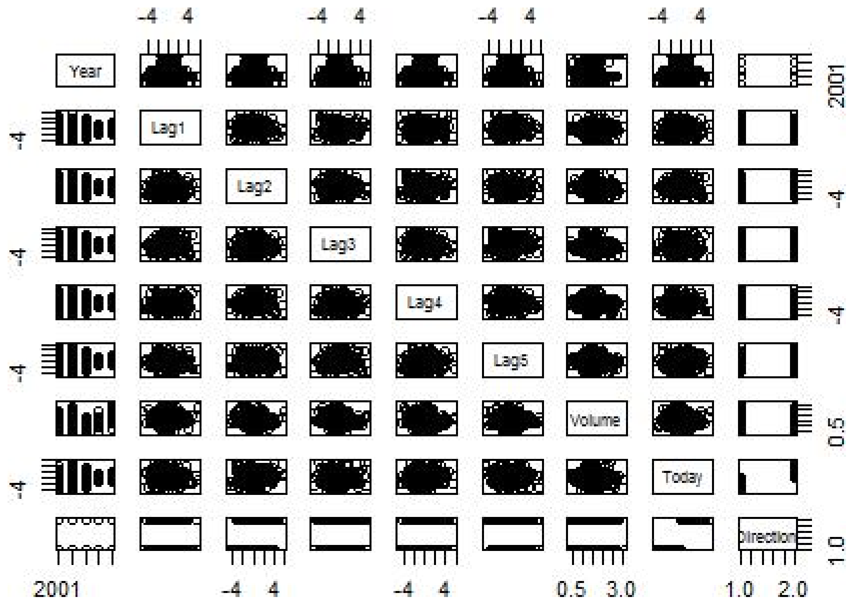


Figure 8.1.5: Scatter Plot Matrix

# 3. CONCLUSION

In conclusion, this research is about image classification by using deep learning via framework TensorFlow. It has three 3 objectives that have achieved throughout this research. The objectives are linked directly with conclusions because it can determine whether all objectives are successfully achieved or not. It can be concluded that all results that have been obtained, showed quite impressive outcomes. The deep neural network (DNN) becomes the main agenda for this research, especially in image classification technology. DNN technique was studied in more details starting from assembling, training model and to classify images into categories. The roles of epochs in DNN was able to control accuracy and also prevent any problems such as overfitting. Implementation of deep learning by using framework TensorFlow also gave good results as it is able to simulate, train and classified with up to 90% percent of accuracy towards nine different types of clothes that have become a trained model.

LDA(Linear Discriminant Analysis) in python is a very simple and well-understood approach to classification in Machine learning. Though there are dimension reduction techniques like LR LDA is preferred in many special classification cases. LDA is a simple prototype classifier. It provides an informative low-dimensional view of the data, which is both useful for visualization and feature engineering. Support for more complex prototype classified is desired.

## 4. REFERENCES

[1] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A Recurrent Neural Network For Image Generation.
https://doi.org/10.1038/nature14236

[2] Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-net: Imagenet classification using binary convolutional neural networks. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9908 LNCS, 525–542.
https://doi.org/10.1007/978-3-319-46493-0_32

[3] Kamavisdar, P., Saluja, S., & Agrawal, S. (2013). A survey on image classification approaches and techniques. Nternational Journal of Advanced Research in Computer and Communication Engineering, 2(1), 1005–1009.
https://doi.org/10.23883/IJRTER.2017.3033.XTS7Z

[4] Pasolli, E., Melgani, F., Tuia, D., Pacifici, F., & Emery, W. J. (2014). SVM active learning approach for image classification using spatial information. IEEE Transactions on Geoscience and Remote Sensing, 52(4), 2217–2223.
https://doi.org/10.1109/TGRS.2013.2258676

[5] Korytkowski, M., Rutkowski, L., & Scherer, R. (2016). Fast image classification by boosting fuzzy classifiers. Information Sciences, 327, 175–182.
https://doi.org/10.1016/j.ins.2015.08.030