# Description of the project

Deepesh Kataria

June 6th 2019

Linear Regression and logistic regressions sound complex terms but algorithms to solve the normal calculations for complex datasets. For example- Every person who do shopping does logistic regression in their daily life. Suppose a mother knows the favorite color of his child or what king of food he likes based on his previous choices or the data she has on his son. It is simplest example of logistic regression she can decide from two things and give the probability of likeability for both products for his son.

We have a dataset that consists of labels from 0-9 in total we have 10 variables to predict and 784 pixels. The predictive variable is our objective variable and the 784 pixels are the independent variables which can help us to predict the label.

Logistic Regression-

Label – a*Pixel1 + pixel2 +………… +ag* Pixel784

This is a equation which gives the probability of the label and by using sigmoid function we can categories the label into 1 and 0. We can use the optimal cutoff in r which will provide the cutoff of probability based on the training dataset but that can result in model overfitting.

**Software used** – Jupiter Notebook (Anaconda 3), Pycharm

**Approaches** used to predict the label on the test dataset

- Taking the 20000 random sample and then training 10 models corresponding to every label and then finding the probabilities to of the labels. Label with highest probability for the pixels will be the predicted label.
- Using the logistic regression on the whole training dataset and checking the accuracy – *Source 2*
- Using the multinomial logistic regression with softmax function – *Source 2*

We used the following libraries in python

- Pandas for data analysis
- Numpy for the numerical python
- Matplotlib for plotting
- Math for mathematics
- Sklearn for the logistic modelling

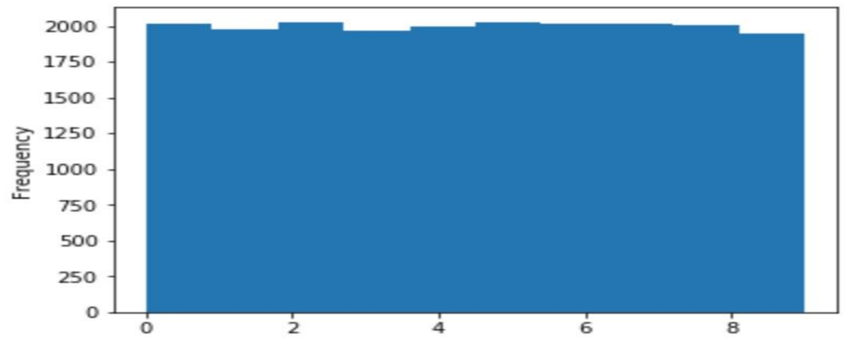
Step for Approach 1 –

We’ll import the data into the Jupiter or pycharm using the pandas package and then select the random 20000 fields from the training dataset.

```
training = pd.read_csv("C:/Users/Deepesh/Desktop/fashion_train.csv", header=None)
test = pd.read_csv("C:/Users/Deepesh/Desktop/fashion_test.csv", header=None)
training.head(10)
training_i = training.sample(frac=1) tr_i
= training_i.iloc[0:20000,:]
```

Checking the distribution of sample data

```
tr_i["label"].plot.hist()
<matplotlib.axes._subplots.AxesSubplot at 0x25645722240>
```



The data is almost equally distributed.

Then we'll rename the label and assign 1 to the objective label and 0 to the rest of the labels

```
tr_i.rename(columns={0:'label'},inplace=True) tr_i_9 =
tr_i
tr_i_9['label'] = np.where(tr_i_9['label'] < 9, 0, 1)
test_9 = test
test_9['label'] = np.where(test_9['label'] < 9, 0, 1)
```

Then we'll separate the objective variable and dependent variables.

```
X_9=tr_i_9.drop("label",axis=1)
Y_9=tr_i_9["label"]
X_test_9 = test_9.drop("label",axis=1)
Y_test_9= test_9["label"]
```

Then we'll import the linear model from the sklearn and set the model as logistic regression. We'll train the model using the X and Y dataframes.

```
from sklearn import linear_model lm =
linear_model.LogisticRegression()
model_9 = lm.fit(X_9, Y_9)
predictions_9 = model_9.predict(X_test_9)
```

Then we can predict the values use confusion matrix for our reference and calculate the accuracy of the model for each label just to check and then we can predict the probability array of the label.

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test_9,predictions_9) from sklearn.metrics
```

```
import accuracy_score accuracy_score(Y_test_9,predictions_9)
probability_9 = lm.predict_proba(X_test_9)
```

Then we'll save the results of the probability in a dataframe using numpy from the array.

```
probability9 =pd.DataFrame(probability_9,index=probability_9[:,0])
```

****** Repeated the steps for every label and get the probability data frames for every label******

Then we'll get the probabilities of 0 and 1 for each label then we'll separate the probabilities of 1 for each label

Code:

```
prob5 =pd.DataFrame(probability_5, columns =list('05'))
prob6 =pd.DataFrame(probability_6, columns =list('06'))
prob7 =pd.DataFrame(probability_7, columns =list('07'))
prob8 =pd.DataFrame(probability_8, columns =list('08'))
prob9 =pd.DataFrame(probability_9, columns =list('09'))
prob0 =pd.DataFrame(probability_0, columns =list('c0'))
prob1 =pd.DataFrame(probability_1, columns =list('01'))
prob2 =pd.DataFrame(probability_2, columns =list('02'))
prob3 =pd.DataFrame(probability_3, columns =list('03'))
prob4 =pd.DataFrame(probability_4, columns =list('04'))
prob0 =prob0.iloc[:,1:2] prob1 =prob1.iloc[:,1:2] prob2
=prob2.iloc[:,1:2] prob3 =prob3.iloc[:,1:2] prob4
=prob4.iloc[:,1:2] prob5 =prob5.iloc[:,1:2] prob6
=prob6.iloc[:,1:2] prob7 =prob7.iloc[:,1:2] prob8
=prob8.iloc[:,1:2] prob9 =prob9.iloc[:,1:2]
```

Then we'll append the probabilities of all the labels into one dataframe.

```
prob_t = prob0.join(prob1) prob_t_2_3
= prob2.join(prob3) prob_t_4_5 =
prob4.join(prob5) prob_t_6_7 =
prob6.join(prob7) prob_t_8_9 =
prob8.join(prob9)
prob_2_3_4_5 = prob_t_2_3.join(prob_t_4_5)
prob_6_7_8_9= prob_t_6_7.join(prob_t_8_9)
prob_not_0= prob_2_3_4_5.join(prob_6_7_8_9) prob_all
= prob_t.join(prob_not_0)
```

Then we'll apply a For loop to get the maximum probability label and those are the predicted labels for the test datasets

```
#Getting label with max probability temp
= prob_all.idxmax(axis=1)
```

This code will get the column index for the maximum value of the row which is the predicted label but this will give us a dataframe with object type then we'll change the type to int.

```
#Changing type to int
Temp_f=temp.astype(int)
```

Now we'll get the confusion matrix and accuracy of the model.

```
#Plotting the confusion mattrix
test_F= test.iloc[0:5000,:] Y_Test_f
= test_F["label"]
confusion_matrix(Y_Test_f,Temp_f)

#Getting the Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(Y_Test_f,Temp_f)
```

**Code for Approach 2 – Using Softmax function instead of sigmoid**

```
#Mulinomial model
training_m = pd.read_csv("C:/Users/Deepesh/Desktop/fashion_train.csv", header=None)
training_m_r = training_m.sample(frac=1) t_m = training_m_r.iloc[0:20000,:]
t_m.rename(columns={0:'label'},inplace=True)
X_m=t_m.drop("label",axis=1)
Y_m=t_m["label"]

#Model
mul_lr = linear_model.LogisticRegression(multi_class='multinomial',
solver='newtoncg').fit(X_m, Y_m)
test_m = pd.read_csv("C:/Users/Deepesh/Desktop/fashion_test.csv", header=None)
test_m.rename(columns={0:'label'},inplace=True)
Xtest_m=test_m.drop("label",axis=1)
Ytest_m=test_m["label"]
#Predictions
predictions = mul_lr.predict(Xtest_m)
#Confusion matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(Ytest_m,predictions)
#Accuracy
from sklearn.metrics import accuracy_score
accuracy_score(Ytest_m,predictions)
```

**Code for Approach 3- Multinomial logistic regression**

```
# Train multi-classification model with logistic
regression lr = linear_model.LogisticRegression()
lr.fit(X_m, Y_m)
predictions_l = lr.predict(Xtest_m)
#Confusion MAttrix
confusion_matrix(Ytest_m,predictions_l)
#Accuracy
accuracy_score(Ytest_m,predictions_l)
```

**Results:**

Confusion Matrix from the 3 Models

**Assembled logistic regression probabilities confusion matrix-**

| | | Predicted Label | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Label | 0 | 397 | 3 | 12 | 27 | 8 | 0 | 46 | 0 | 13 | 1 |
| | 1 | 2 | 457 | 1 | 13 | 4 | 0 | 3 | 0 | 1 | 0 |
| | 2 | 7 | 11 | 368 | 8 | 75 | 0 | 44 | 0 | 8 | 0 |
| | 3 | 18 | 26 | 13 | 400 | 16 | 0 | 11 | 1 | 11 | 4 |
| | 4 | 4 | 5 | 57 | 23 | 380 | 0 | 40 | 0 | 12 | 0 |
| | 5 | 1 | 4 | 1 | 7 | 2 | 418 | 2 | 27 | 4 | 19 |
| | 6 | 68 | 7 | 64 | 19 | 50 | 0 | 253 | 0 | 20 | 1 |
| | 7 | 0 | 0 | 1 | 2 | 0 | 17 | 0 | 462 | 0 | 18 |
| | 8 | 5 | 6 | 6 | 8 | 3 | 9 | 15 | 4 | 467 | 3 |
| | 9 | 1 | 0 | 1 | 1 | 0 | 14 | 3 | 22 | 4 | 431 |

As we can see in the confusion matrix most of the predictions are along the diagonal that means that the model is good. The predictions of 5 is done with a great accuracy with error allocation of 40 out of 498 and the predictions for 4 has more of the dislocation with total number of 158 out of 696.

**Confusion matrix for Multinomial model**

| | | Predicted Label | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Label | 0 | 741 | 24 | 23 | 56 | 10 | 2 | 119 | 0 | 24 | 1 |
| | 1 | 6 | 947 | 9 | 24 | 8 | 0 | 4 | 0 | 2 | 0 |
| | 2 | 30 | 15 | 691 | 19 | 118 | 1 | 111 | 0 | 15 | 0 |
| | 3 | 58 | 42 | 25 | 776 | 43 | 0 | 38 | 0 | 18 | 0 |
| | 4 | 8 | 13 | 141 | 42 | 696 | 2 | 85 | 0 | 12 | 1 |
| | 5 | 0 | 2 | 1 | 2 | 0 | 845 | 0 | 59 | 15 | 76 |
| | 6 | 131 | 18 | 121 | 59 | 115 | 3 | 521 | 1 | 29 | 2 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 53 | 0 | 900 | 3 | 44 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 8 | 16 | 4 | 27 | 15 | 14 | 13 | 26 | 9 | 873 | 3 |
| | 9 | 0 | 0 | 0 | 0 | 0 | 39 | 1 | 47 | 2 | 911 |

In this matrix the accurate allocation is maximum for 7 with 106 out of 1016 and the largest error in allocation is for 6 with 384 misallocations out of 905.

**Confusion matrix for logistic regression on multiclass variables**

| | | Predicted Label | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Actual Label | 0 | 762 | 24 | 16 | 54 | 13 | 3 | 107 | 0 | 19 | 2 |
| | 1 | 8 | 948 | 7 | 22 | 7 | 0 | 4 | 1 | 3 | 0 |
| | 2 | 32 | 19 | 725 | 14 | 122 | 1 | 78 | 0 | 8 | 1 |
| | 3 | 40 | 43 | 13 | 799 | 37 | 6 | 34 | 0 | 16 | 12 |
| | 4 | 2 | 13 | 130 | 50 | 715 | 0 | 78 | 0 | 12 | 0 |
| | 5 | 3 | 4 | 7 | 2 | 0 | 844 | 21 | 43 | 16 | 60 |
| | 6 | 147 | 21 | 128 | 49 | 117 | 1 | 499 | 0 | 37 | 1 |
| | 7 | 0 | 1 | 1 | 1 | 1 | 49 | 6 | 890 | 0 | 51 |
| | 8 | 10 | 7 | 20 | 11 | 6 | 19 | 34 | 11 | 877 | 5 |
| | 9 | 5 | 0 | 7 | 1 | 1 | 28 | 22 | 45 | 1 | 890 |

In this matrix the accurate allocation is also maximum for 7 with 100 out of 990 and largest error in allocation is for 6 with 384 out of 883.

**Findings of confusion matrices**

- The most accurate allocated label and label with lowest accuracy allocations are adjacent ((5,4),(7,6),(7,6))
- Multinomial model and Logistic regression model with multiclass variables have the similar position of error that shows that the models are not over fitted on the data

## Accuracy

| Model | Accuracy |
|---|---|
| Assembled probability | 81% |
| Multinomial model | 79% |
| Logistic regression Multiclass | 79.47% |

The largest accuracy is achieved by the assembled probability of the logistic regression's models for each label. Though we have used the logistic regression model directly on the data provided the different accuracy rate. The reason could be the variety of the test set we've used in both models. For, the first model we've only considered the 5000 test values 25% of the training set but for third model we've used the all test values. Multinomial model and logistic regression for multiclass variable showed the almost same accuracy and same error position(as seen in the confusion matrix)

## References

**Stack Overflow – For Code Help-** https://stackoverflow.com/

Source 2 - https://dataaspirant.com/2017/05/15/implement-multinomial-logistic-regression-python/