National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Sayan Ghosh
Student ID: 21143838

School of Computing
National College of Ireland

Supervisor: Dr. Cristina Hava Muntean

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sayan Ghosh		
Student ID:	X21143838		
Programme:	Data Analytics	**Year:**	2022-23
Module:	MSc Research Project		
Lecturer:	Dr. Cristina Hava Muntean		
Submission Due Date:	15/12/2022		
Project Title:	Application of Data Analytics to Suggest Gifts to Retail Customers Based on their Emotions		
Word Count:	**797**	**Page Count: 12**	

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sayan Ghosh
Student ID: 21143838

1 Introduction

This configuration handbook provides information on the system configuration, software, and hardware requirements, as well as the steps necessary to complete the Research Project: Application of Data Analytics to Suggest Gifts to Retail Customers Based on their Emotions.

Software and hardware specifications are covered in Section 2 of this document. The environment setup, data collection, data preparation, and the libraries to be imported are covered in Section 3. The different steps taken for the image processing dataset are explained in Section 4. In Section 5, the model design for both the image dataset and the gift dataset is discussed. In the last section (Section 6) all the steps taken for the final Testing and Deployment stage have been shown.

2 System Configuration

The hardware and Software specifications required for this project have mentioned below in this section.

2.1 Hardware Requirements

Table 1: Hardware Requirements

Operating System	Windows 10
RAM	8 GB
Hard Disc Space	250 GB
CPU	64 bit x 86 multi-core

2.2 Soft Requirements

Table 2: Software Requirements

Programming Language Tools	Jupyter Notebook
Web Browser	Google Chrome or Mozilla Firefox
Other Software	Microsoft Excel, Microsoft Word, and Overleaf

3 Project Development

In this section, information related to the environment setup, data collection, data preparation, and the libraries to be imported are covered.

3.1 Environmental Setup

Jupyter Notebook has been used to carry out the implementation part of the project.

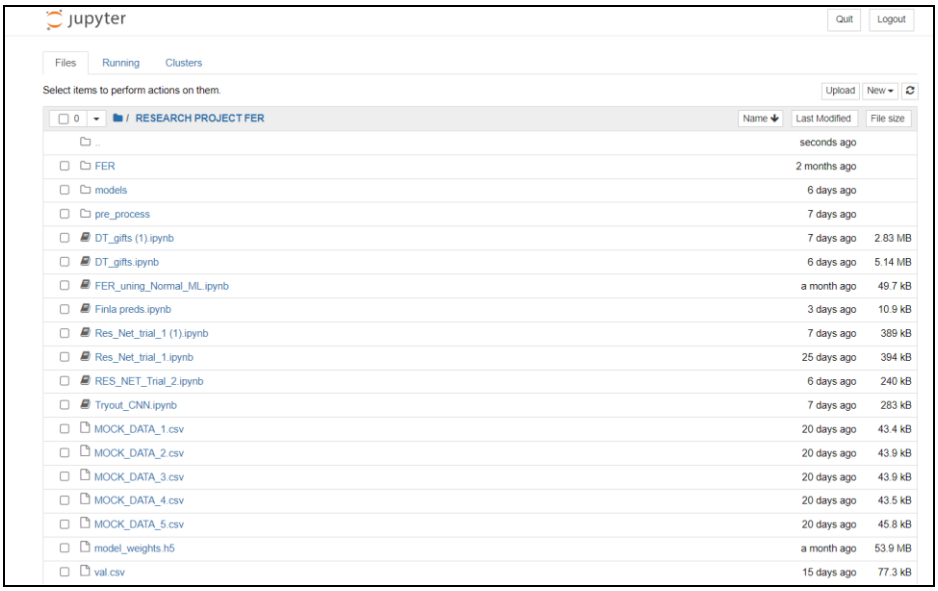

Figure 1: Jupyter Notebook

3.2 Data Collection

The Image dataset used in this project is FER2013. The dataset has been taken from Kaggle1. It is a reliable data source that is also open to the public. In the form of ".jpg" files, image data is collected.

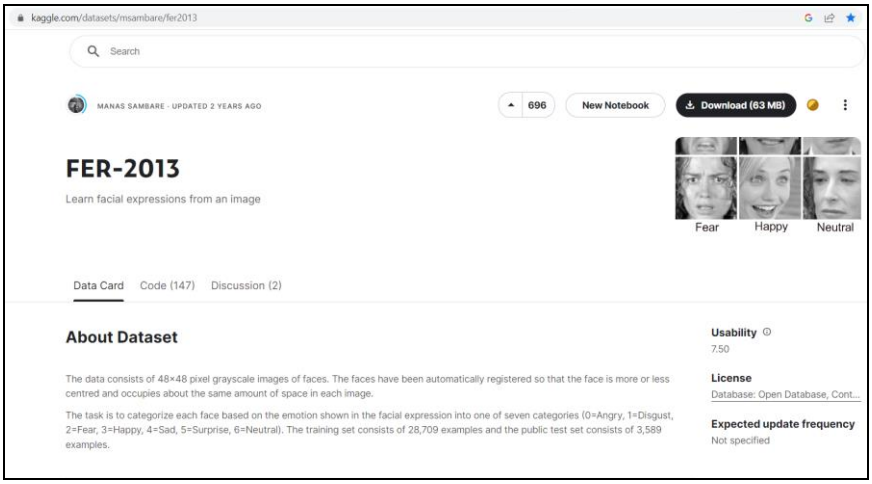

Figure 2: Data Source- Kaggle

[1] https://www.kaggle.com/datasets/msambare/fer2013

3.3 Data Preparation

The method used to prepare the data is described in this section. Training, Validation, and Testing are the three folders into which the data is separated and saved in the directory.

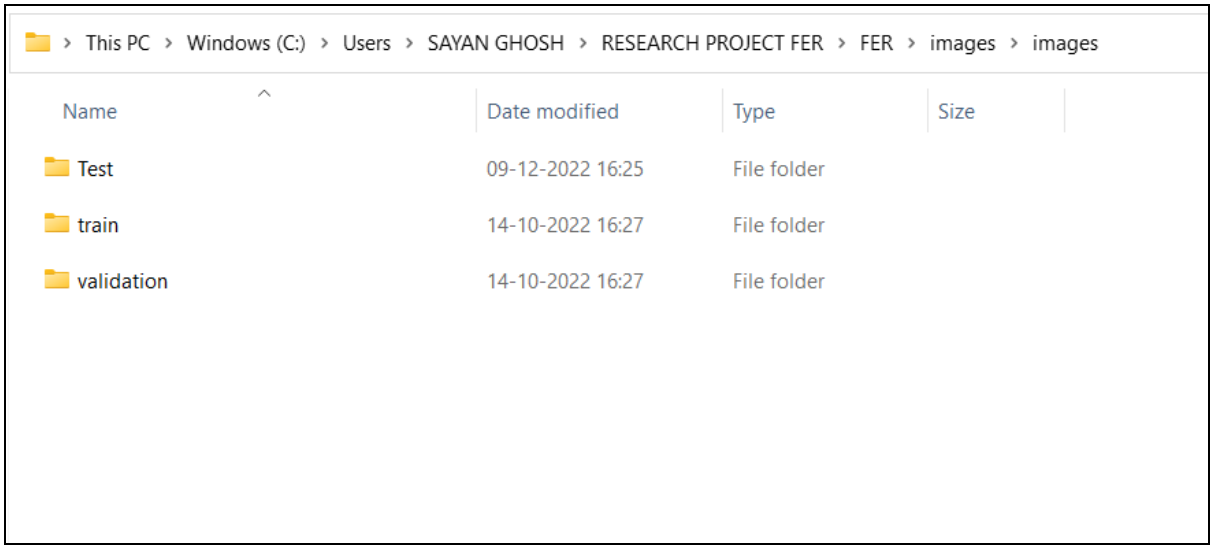

Figure 3: Data in Directory

3.4 Importing Libraries

The main libraries required for this project are NumPy, Seaborn, Matplotlib, Tensorflow, Keras, sklearn, and Pandas.

```
import numpy as np
import seaborn as sns
# from keras.preprocessing.image import  img_to_array
from tensorflow.keras.utils import load_img,img_to_array
import matplotlib.pyplot as plt
import os
```

```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np
import os
import pprint
import pandas as pd
from sklearn.preprocessing import LabelEncoder
# pp = pprint.PrettyPrinter(indent=4)
```

Figure 4: Imported Libraries

4 Image Processing

All the steps taken for pre-processing and organizing the image data are shown in this section.

4.1 Pre-processing and augmentation for ResNet50 Model

```
# building data generator

from keras.preprocessing.image import ImageDataGenerator

batch_size = 128
base_path = r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\FER\images\images'


train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                   width_shift_range = 0.1,
                                    height_shift_range = 0.1,
                                    rotation_range = 20,
                                    horizontal_flip = True)

validation_datagen = ImageDataGenerator(rescale= 1.0/255)

train_generator = train_datagen.flow_from_directory(base_path + "/train",
                                                    target_size=(56,56),
                                                    color_mode="rgb",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(base_path + "/validation",
                                                    target_size=(56,56),
                                                    color_mode="rgb",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=False)
```

```
Found 28821 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

```
from collections import Counter
counter = Counter(train_generator.classes)
max_val = float(max(counter.values()))
class_weights = {class_id : max_val/num_images for class_id, num_images in counter.items()}
```

Figure 5: Pre-processing for ResNet50 Model

4.2 Pre-processing and image augmentation for Convolutional Neural Network (CNN) Model

Image augmentation using keras ImageDataGenerator

```
# building data generator

from keras.preprocessing.image import ImageDataGenerator

batch_size = 128
base_path = r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\FER\images\images'


train_datagen = ImageDataGenerator(rescale = 1.0/255.0,
                                   width_shift_range = 0.1,
                                    height_shift_range = 0.1,
                                    rotation_range = 20,
                                    horizontal_flip = True)

validation_datagen = ImageDataGenerator(rescale= 1.0/255)

train_generator = train_datagen.flow_from_directory(base_path + "/train",
                                                    target_size=(56,56),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(base_path + "/validation",
                                                    target_size=(56,56),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=False)
```

```
Found 28821 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

Figure 6: Pre-processing for CNN Model

4.3 Pre-processing and image augmentation for Random Forest Model (Used for Image Processing)

```
import joblib
from skimage.io import imread
from skimage.transform import resize

def resize_all(src,  include, width=150, height=None):
    """
    load images from path, resize them and write them as arrays to a dictionary,
    together with labels and metadata. The dictionary is written to a pickle file
    named '{pklname}_{width}x{height}px.pkl'.

    Parameter
    ---------
    src: str
        path to data
    pklname: str
        path to output file
    width: int
        target width of the image in pixels
    include: set[str]
        set containing str
    """

    height = height if height is not None else width

    data = dict()
    data['description'] = 'resized ({0}x{1})FER images in grayscale'.format(int(width), int(height))
    data['label'] = []
    data['filename'] = []
    data['data'] = []

#     pklname = f"{pklname}_{width}x{height}px.pkl"

    # read all images in PATH, resize and write to DESTINATION_PATH
    for subdir in os.listdir(src):
        if subdir in include:
            print(subdir)
            current_path = os.path.join(src, subdir)

            for file in os.listdir(current_path):
#                 if file[-3:] in {'jpg', 'png'}:
                im = imread(os.path.join(current_path, file))
                im = resize(im, (width, height)) #[:,:,::-1]
                data['label'].append(subdir)
                data['filename'].append(file)
                data['data'].append(im.flatten())
    return data
#         joblib.dump(data, pklname)
```

```
data_path = r'C:\\Users\\SAYAN GHOSH\\RESEARCH PROJECT FER\\FER\\images\\images\\train'
os.listdir(data_path)
```

```
['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise']
```

```
# base_name = 'emots'
width = 80

include = {'angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'}

data = resize_all(src=data_path, width=width, include=include)
```

```
angry
disgust
fear
happy
neutral
sad
surprise
```

```
# base_name = 'emotions'
val_path = r'C:\\Users\\SAYAN GHOSH\\RESEARCH PROJECT FER\\FER\\images\\images\\validation'
width = 80

include = {'angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surprise'}

data_val = resize_all(src=val_path, width=width, include=include)
```

```
from collections import Counter

# data = joblib.load(f'{base_name}_{width}x{width}px.pkl')

print('number of samples: ', len(data['data']))
print('keys: ', list(data.keys()))
print('description: ', data['description'])
print('image shape: ', data['data'][0].shape)
print('labels:', np.unique(data['label']))

Counter(data['label'])
```

```
number of samples:  28821
keys:  ['description', 'label', 'filename', 'data']
description:  resized (80x80)FER images in grayscale
image shape:  (6400,)
labels: ['angry' 'disgust' 'fear' 'happy' 'neutral' 'sad' 'surprise']

Counter({'angry': 3993,
         'disgust': 436,
         'fear': 4103,
         'happy': 7164,
         'neutral': 4982,
         'sad': 4938,
         'surprise': 3205})
```

```
X = np.array(data['data'])
y = np.array(data['label'])
```

```
# data['label']
X_val = np.array(data_val['data'])
y_val = np.array(data_val['label'])
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    shuffle=True,
    random_state=69,
)
```

```
from sklearn.base import BaseEstimator, TransformerMixin

class RGB2GrayTransformer(BaseEstimator, TransformerMixin):
    """
    Convert an array of RGB images to grayscale
    """

    def __init__(self):
        pass

    def fit(self, X, y=None):
        """returns itself"""
        return self

    def transform(self, X, y=None):
        """perform the transformation and return an array"""
        return np.array([skimage.color.rgb2gray(img) for img in X])
```

Figure 7: Pre-processing for Random Forest Model

4.4 Pre-processing for the Gift dataset

```
from sklearn.preprocessing import OneHotEncoder
```

```
y_train = df['Gift Categories']
y_test = df_test['Gift Categories']
```

```
OHE = OneHotEncoder(
    categories='auto',  # Categories per feature
    drop=None, # Whether to drop one of the features
    sparse=True, # Will return sparse matrix if set True
    handle_unknown='error' # Whether to raise an error
)
OHE.fit(df[['Gender']])
transformed = OHE.transform(df[['Gender']])
transformed_tst = OHE.transform(df_test[['Gender']])
df[OHE.categories_[0]] = transformed.toarray()
df_test[OHE.categories_[0]] = transformed_tst.toarray()
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\OHE_Gend', "wb") as f:
    pickle.dump(OHE,f)
```

```
OHE = OneHotEncoder(
    categories='auto',  # Categories per feature
    drop=None, # Whether to drop one of the features
    sparse=True, # Will return sparse matrix if set True
    handle_unknown='error' # Whether to raise an error
)
OHE.fit(df[['Occasion']])
transformed = OHE.transform(df[['Occasion']])
transformed_tst = OHE.transform(df_test[['Occasion']])
df[OHE.categories_[0]] = transformed.toarray()
df_test[OHE.categories_[0]] = transformed_tst.toarray()
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\OHE_OCC', "wb") as f:
    pickle.dump(OHE,f)
```

```
le = LabelEncoder()
le.fit(df['Emotions'])
# le.classes_
df['Emotions'] = le.transform(df['Emotions'])
df_test['Emotions'] = le.transform(df_test['Emotions'])
df['Emotions'] = df.Emotions.astype('category')
df_test['Emotions'] = df_test.Emotions.astype('category')
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\le_Emo', "wb") as f:
    pickle.dump(le,f)
```

```
le_y = LabelEncoder()
le_y.fit(y_train)
y_train = le_y.transform(y_train)
y_test = le_y.transform(y_test)
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\le_op', "wb") as f:
    pickle.dump(le_y,f)
```

```
df.drop(columns=['Gender','Occasion','Age','Gift Categories'],inplace=True)
df_test.drop(columns=['Gender','Occasion','Age','Gift Categories'],inplace=True)
```

```
df.dtypes
```

```
Emotions          category
Price (Euros)        int64
Female             float64
Male               float64
Casual             float64
Festivals          float64
Special Events     float64
dtype: object
```

Figure 8: Pre-processing for the Gift Dataset

5 Model Design

5.1 Modelling for Image Dataset

5.1.1 ResNet50 Model

```
# Create and compile model
from tensorflow.keras.applications import ResNet50

from keras.models import Model, Sequential
from keras.layers import Dense, Input, Dropout, GlobalMaxPooling2D, Flatten, Conv2D, BatchNormalization, Activation, MaxPooling2D
from keras.optimizers import Adam
model = Sequential()
model.add(ResNet50(input_shape=(56, 56, 3), weights='imagenet', =False))
model.add(GlobalMaxPooling2D())
model.add(Dropout(0.5))
model.add(Dense(7,activation='softmax'))
```

```
from tensorflow.keras.utils import plot_model
model.compile(optimizer=Adam(learning_rate=1e-4), loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
history = model.fit(train_generator,
                    validation_data = validation_generator,
                    class_weight=class_weights,
                    epochs = 35,
                    steps_per_epoch=train_generator.n//train_generator.batch_size,
                   validation_steps = validation_generator.n//validation_generator.batch_size)
```

Figure 9: Model Building for ResNet50

5.1.2 CNN Model

```
from keras.layers import Dense, Input, Dropout, GlobalAveragePooling2D, Flatten, Conv2D, BatchNormalization, Activation, MaxPool
from keras.models import Model, Sequential
from keras.optimizers import Adam

# number of possible label values
nb_classes = 7

# Initialising the CNN
model = Sequential()

# 1 - Convolution
model.add(Conv2D(64,(3,3), padding='same', input_shape=(56, 56,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 2nd Convolution layer
model.add(Conv2D(128,(5,5), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 3rd Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 4th Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(nb_classes, activation='softmax'))

print(model.summary())

opt = Adam(learning_rate=0.0001)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

```
epochs = 35

# checkpoint to save best model
from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

history = model.fit_generator(generator=train_generator,
                              steps_per_epoch=train_generator.n//train_generator.batch_size,
                              epochs=epochs,
                              validation_data = validation_generator,
                              validation_steps = validation_generator.n//validation_generator.batch_size,
                              callbacks=callbacks_list
                              )
```

Figure 10: Model Building for CNN

5.1.3 Random Forest Model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_predict
from sklearn.preprocessing import StandardScaler, Normalizer
import skimage

# create an instance of each transformer
grayify = RGB2GrayTransformer()

rfc = RandomForestClassifier(n_estimators = 300)

# call fit_transform on each transform converting X_train step by step
# X_train_gray = grayify.fit_transform(X_train_1)
scalify = StandardScaler()
X_train_prepared = scalify.fit_transform(X_train)
rfc.fit(X_train_prepared, y_train)
# print(X_train_prepared.shape)
```

Figure 11: Model Building for Random Forest (Image processing)

5.2 Modelling for Gift Dataset

5.2.1 Random forest Model

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 500,random_state = 69)
rfc.fit(df, y_train)
```

```
RandomForestClassifier(n_estimators=500, random_state=69)
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\models\RandF', "wb") as f:
    pickle.dump(rfc,f)
```

```
y_pred = rfc.predict(df)
print('Train Acc: ', 100*np.sum(y_pred == y_train)/len(y_train))
y_pred = rfc.predict(df_test)
print('Validation Acc: ', 100*np.sum(y_pred == y_test)/len(y_test))
```

Figure 12: Model Building for Random Forest (Gift Dataset)

5.2.2 Decision Tree Model

Trying Decission Tree

```
clf = DecisionTreeClassifier(random_state = 70)
# max_depth =10
clf.fit(df, y_train)
```

```
DecisionTreeClassifier(random_state=70)
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\models\DTree', "wb") as f:
    pickle.dump(clf,f)
```

```
from sklearn import tree

import matplotlib.pyplot as plt

plt.figure(figsize=(50,60))

a = tree.plot_tree(clf,

                   feature_names = df.columns,
                   rounded = True,
                   filled = True,
                   fontsize=14)

plt.show()
```

```
y_pred = clf.predict(df)
print('Train Acc: ', 100*np.sum(y_pred == y_train)/len(y_train))
y_pred = clf.predict(df_test)
print('Validation Acc: ', 100*np.sum(y_pred == y_test)/len(y_test))
```

Figure 13: Model Building for Decision Tree

5.2.3 Logistic Regression Model

Trying logistic regression

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```

```
LR = LogisticRegression(multi_class= 'multinomial',solver = 'newton-cg')
LR.fit(df, y_train)
```

```
LogisticRegression(multi_class='multinomial', solver='newton-cg')
```

```
import pickle
with open(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\models\Logit', "wb") as f:
    pickle.dump(LR,f)
```

```
y_pred = LR.predict(df)
print('Train Acc: ', 100*np.sum(y_pred == y_train)/len(y_train))
y_pred = LR.predict(df_test)
print('Validation Acc: ', 100*np.sum(y_pred == y_test)/len(y_test))
```

Figure 14: Model Building for Logistic Regression

6 Testing & Deployment

```
def DL_Pred(batch_size = 128, path = r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\FER\images\images\\',model_path = r'C:\Users\SA
    bat_size = batch_size
    base_path = path
    test_datagen = ImageDataGenerator(rescale= 1.0/255)
    test_generator = test_datagen.flow_from_directory(base_path + "Test",
                                                      target_size=(56,56),
                                                      color_mode="rgb",
                                                      batch_size=batch_size,
                                                      class_mode='categorical',
                                                      shuffle=False)
    # label_map
    label_map = {0: 'Angry',1: 'Disgust',2: 'Fear',3: 'Happy',4: 'Neutral',5: 'Sad',6: 'Surprise'}
    RNN = DL_model(model_path)
    preds = RNN.predict(test_generator)
    pp = preds.argmax(axis=-1)
    cls = []
    for i in pp:
        cls.append(label_map[i])
    return cls
```

```
def Pickle_file_loader(file_name):
    file = open(file_name,"rb")
    pkl = pickle.load(file)
    return pkl
```

```
def gift_data_prep(Price, Gender, Occasion):
    df = pd.DataFrame({'Price (Euros)':Price.split(','), 'Gender':Gender.split(','), 'Occasion':Occasion.split(','), 'Emotions':[
    OHE_Gend = Pickle_file_loader(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\OHE_Gend')
    transformed = OHE_Gend.transform(df[['Gender']])
    df[OHE_Gend.categories_[0]] = transformed.toarray()
    OHE_OCC = Pickle_file_loader(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\OHE_OCC')
    transformed = OHE_OCC.transform(df[['Occasion']])
    df[OHE_OCC.categories_[0]] = transformed.toarray()
    le_Emo = Pickle_file_loader(r'C:\Users\SAYAN GHOSH\RESEARCH PROJECT FER\pre_process\le_Emo')
    df['Emotions'] = le_Emo.transform(df[['Emotions']])
    df['Emotions'] = df.Emotions.astype('category')
    df['Price (Euros)'] = df['Price (Euros)'].astype('int')
    df.drop(columns=['Gender','Occasion'],inplace=True)
    return df
```

```
gift_data_prep(Price = input('Enter Price:'), Gender = input("Enter Gender (Male\Female):"), Occasion = input("Enter Occasion:"))
```

Figure 15: Testing & Deployment