```
**Project title** : Intelligent Vehicle Damage Assessment and Cos
Estimator for Insurance Companies
<br>
**Name**: Gowtham P<br>
**Roll Number**:720719104063
```

Project title : Intelligent Vehicle Damage Assessment and Cost Estimator for Insurance Companies
Name: Gowtham P
Roll Number:720719104063

Import the data

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,
                                 zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Image Augmentation

```
x_train = train_datagen.flow_from_directory('/content/gdrive/MyDrive/flowers',
                                       target_size=(64,64),
                                       class_mode='categorical',
                                       batch_size=100)
```

```
Found 4317 images belonging to 5 classes.
```

```
x_test=test_datagen.flow_from_directory("/content/gdrive/MyDrive/flowers",target_size=(64,64),class_mode='categorical',batch_size=
```

```
Found 4317 images belonging to 5 classes.
```

```
x_train.class_indices
```

```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Dense, Flatten
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
```

INITIALISING AND CREATING MODEL

```
model = Sequential()
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3)))
model.add(MaxPooling2D((2,2)))
model.add(Flatten())
model.add(Dense(300,activation='relu'))
model.add(Dense(150,activation='relu'))
model.add(Dense(5,activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 flatten (Flatten)           (None, 30752)             0

 dense (Dense)               (None, 300)               9225900
 dense_1 (Dense)             (None, 150)               45150

 dense_2 (Dense)             (None, 5)                 755

=================================================================
Total params: 9,272,701
Trainable params: 9,272,701
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit_generator(x_train,steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=len(x_test), epochs= 30)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is deprecated and will
  """Entry point for launching an IPython kernel.
Epoch 1/30
44/44 [==============================] - 1198s 27s/step - loss: 1.7670 - accuracy: 0.3574 - val_loss: 1.1815 - val_accurac
Epoch 2/30
44/44 [==============================] - 54s 1s/step - loss: 1.1316 - accuracy: 0.5363 - val_loss: 1.1560 - val_accuracy:
Epoch 3/30
44/44 [==============================] - 54s 1s/step - loss: 1.0400 - accuracy: 0.5847 - val_loss: 0.9855 - val_accuracy:
Epoch 4/30
44/44 [==============================] - 54s 1s/step - loss: 0.9853 - accuracy: 0.6145 - val_loss: 0.9951 - val_accuracy:
Epoch 5/30
44/44 [==============================] - 54s 1s/step - loss: 0.9483 - accuracy: 0.6368 - val_loss: 0.9216 - val_accuracy:
Epoch 6/30
44/44 [==============================] - 54s 1s/step - loss: 0.9200 - accuracy: 0.6384 - val_loss: 0.8748 - val_accuracy:
Epoch 7/30
44/44 [==============================] - 53s 1s/step - loss: 0.8649 - accuracy: 0.6699 - val_loss: 0.7901 - val_accuracy:
Epoch 8/30
44/44 [==============================] - 54s 1s/step - loss: 0.8380 - accuracy: 0.6762 - val_loss: 0.8823 - val_accuracy:
Epoch 9/30
44/44 [==============================] - 54s 1s/step - loss: 0.7936 - accuracy: 0.6977 - val_loss: 0.8195 - val_accuracy:
Epoch 10/30
44/44 [==============================] - 54s 1s/step - loss: 0.7702 - accuracy: 0.7088 - val_loss: 0.7886 - val_accuracy:
Epoch 11/30
44/44 [==============================] - 54s 1s/step - loss: 0.7270 - accuracy: 0.7250 - val_loss: 0.7067 - val_accuracy:
Epoch 12/30
44/44 [==============================] - 54s 1s/step - loss: 0.7062 - accuracy: 0.7299 - val_loss: 0.7277 - val_accuracy:
Epoch 13/30
44/44 [==============================] - 54s 1s/step - loss: 0.7207 - accuracy: 0.7199 - val_loss: 0.6623 - val_accuracy:
Epoch 14/30
44/44 [==============================] - 54s 1s/step - loss: 0.6783 - accuracy: 0.7452 - val_loss: 0.6552 - val_accuracy:
Epoch 15/30
44/44 [==============================] - 54s 1s/step - loss: 0.6291 - accuracy: 0.7649 - val_loss: 0.5923 - val_accuracy:
Epoch 16/30
44/44 [==============================] - 54s 1s/step - loss: 0.6075 - accuracy: 0.7751 - val_loss: 0.6215 - val_accuracy:
Epoch 17/30
44/44 [==============================] - 55s 1s/step - loss: 0.6105 - accuracy: 0.7700 - val_loss: 0.5874 - val_accuracy:
Epoch 18/30
44/44 [==============================] - 55s 1s/step - loss: 0.6032 - accuracy: 0.7748 - val_loss: 0.5833 - val_accuracy:
Epoch 19/30
44/44 [==============================] - 54s 1s/step - loss: 0.5641 - accuracy: 0.7918 - val_loss: 0.6373 - val_accuracy:
Epoch 20/30
44/44 [==============================] - 54s 1s/step - loss: 0.5191 - accuracy: 0.8040 - val_loss: 0.4446 - val_accuracy:
Epoch 21/30
44/44 [==============================] - 54s 1s/step - loss: 0.5148 - accuracy: 0.8038 - val_loss: 0.4783 - val_accuracy:
Epoch 22/30
44/44 [==============================] - 54s 1s/step - loss: 0.5171 - accuracy: 0.8073 - val_loss: 0.3808 - val_accuracy:
Epoch 23/30
44/44 [==============================] - 55s 1s/step - loss: 0.4879 - accuracy: 0.8221 - val_loss: 0.3838 - val_accuracy:
Epoch 24/30
44/44 [==============================] - 54s 1s/step - loss: 0.4354 - accuracy: 0.8351 - val_loss: 0.3471 - val_accuracy:
Epoch 25/30
44/44 [==============================] - 54s 1s/step - loss: 0.4475 - accuracy: 0.8372 - val_loss: 0.3437 - val_accuracy:
Epoch 26/30
44/44 [==============================] - 54s 1s/step - loss: 0.4662 - accuracy: 0.8307 - val_loss: 0.4308 - val_accuracy:
Epoch 27/30
44/44 [==============================] - 54s 1s/step - loss: 0.4386 - accuracy: 0.8429 - val_loss: 0.4306 - val_accuracy:
Epoch 28/30
```

SAVE THE MODEL

```
# save model
model.save('flowers.h5')
```

Test the model

```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
model=load_model('flowers.h5')
```

```
val = list(x_train.class_indices.keys())
val
```

```
['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

```
img=image.load_img("/content/gdrive/MyDrive/flowers/daisy/2535769822_513be6bbe9.jpg",target_size=(64,64))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
y=np.argmax(model.predict(x),axis=1)

index=['daisy','dandelion','rose','sunflower','tulip']
index[y[0]]
```

```
'daisy'
```