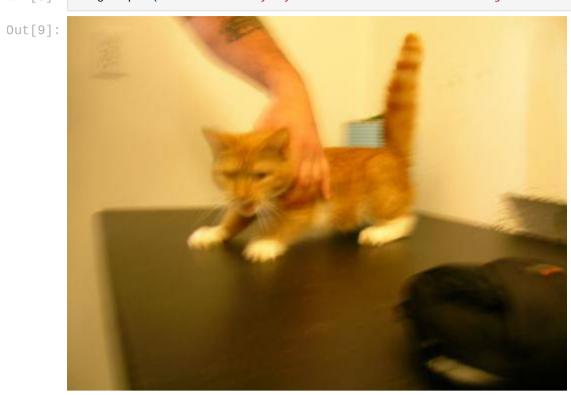
## Project - Image Classification

### Importing libraries

```
In [36]: import os
from skimage.io import imread
from skimage.transform import resize
from PIL import Image
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import *
import warnings
warnings.filterwarnings('ignore')
from sklearn.ree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
```

#### Displaying an image

```
 \label{lem:condition} Image.open(r"C:\Users\Roy\Python files\Submissions\Project - Image-Classification\cats\_and\_dogs\_filtered\train\cats\cat.0.jpg") \\
```



Converting all the images into a 1-D array and storing it into data variable.

Here Label contains two categories i.e 0 (Cat) and 1 (Dog)

```
In [14]: data = []
labels = []

for category_idx, category in enumerate(categories):
    for file in os.listdir(os.path.join(input_dir, category)):
        #loading data
        img_path = os.path.join(input_dir, category, file)
        img = imread(img_path)
        #resizing
        img = resize(img, (15,5))
        data.append(img.flatten())
        labels.append(category_idx)
In [18]: labels = np.asarray(labels)
data = np.asarray(data)
```

## **Logistic Regression**

```
In [20]: X_train, X_test, y_train, y_test = train_test_split(data, labels, train_size=0.8, shuffle=True, stratify=labels)
In [21]: sc = StandardScaler()
    X_train = sc.fit_transform(X_train)
    X_test = sc.transform(X_test)
```

In [24]: lr = LogisticRegression()
lr.fit(X\_train, y\_train)

Out[24]: v LogisticRegression
LogisticRegression()

In [25]: y\_pred = lr.predict(X\_test)
In [26]: accuracy\_score(y\_test, y\_pred)
Out[26]: 0.6325

# Decision Tree

DecisionTreeClassifier()

In [29]: y\_pred = dt.predict(X\_test)

In [30]: accuracy\_score(y\_test, y\_pred)

Out[30]: 0.545

[00].

# Random Forest

```
In [32]: rfc = RandomForestClassifier(n_estimators=1000)
    rfc.fit(X_train, y_train)
```

Out[32]: RandomForestClassifier

RandomForestClassifier(n\_estimators=1000)

In [33]: y\_pred = rfc.predict(X\_test)
accuracy\_score(y\_test, y\_pred)

accuracy\_score(y\_testout[33]:

## Hyperparameter Tuning

```
rfc = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [200,500],
    'max_depth': [4,5,6,7,8],
    'max_depth': [4,5,6,7,8],
    'criterion': ['gini', 'entropy']
}

CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv=5)
CV_rfc.fit(X_train, y_train)
CV_rfc.best_params_
    rfc1 = RandomForestClassifier(random_state=42, max_features='auto', n_estimators=200, max_depth=8, criterion='gini')
    rfc1.fit(X_train, y_train)
pred = rfc1.predict(X_test)
print('Accuracy score: ',accuracy_score(y_test, pred))
```

Accuracy score: 0.635

In [37]: pd.DataFrame({"Actual Value": y\_test, "Predicted Value": y\_pred})

Out[37]:		Actual Value	Predicted Value
	0	0	0
	1	1	0
	2	0	1
	3	0	1
	4	1	0
	395	1	1
	396	0	0
	397	1	1
	398	0	1

400 rows × 2 columns

1