Delicacy Recipe - Classification

Scones? Muffin? or Cupcake? Hi, this time we will try an interesting project: Delicacy.

Classification is a Machine Learning Technique where the Machine Learning Model predicts in a Given Group or a Category i.e Will Rain or Not Rain, Fail or Pass, Pay or Not Pay.

In this Example you will predict in a category of Cakes

This will end up with a model that will automatically predict whether a given recipe lies in the category of Muffin , Scone or a Cupcake?.

Although the cupcake is usually decorated, the shape and appearance are very similar. But the real difference is in the preparation technique. Cupcake is a cup-sized cake. The cupcake recipe uses techniques for preparing the cake dough. And the muffin is considered a quick roll.
Classification is a Supervised Machine Learning Technique where we Train and Test the Models

In Machine Learning these principles apply:

Step 1# Gathering Data

Alice Zhao, a data scientist, did a research and found data by googling 'basic muffin recipe' and 'basic cupcake recipe'.

She found, surprisingly, that muffins, scones and cupcakes just have eight ingredients in them.

At end of the day this is what the data looks like:

```
import pandas
data = pandas.read_csv("http://modcom.co.ke/datasets/data.csv")
data
```

18	36	16	24	12	9	1	1	0	Cupcake
19	34	17	23	11	13	0	1	0	Cupcake
20	29	3	46	6	0	11	0	6	Scone
21	19	3	36	3	14	7	14	5	Scone
22	20	5	52	5	10	3	0	5	Scone
23	19	6	50	5	6	10	0	3	Scone
24	19	10	51	5	0	10	0	5	Scone
25	55	28	3	7	5	2	0	0	Muffin
26	47	24	12	6	9	1	0	0	Muffin
27	47	23	18	6	4	1	0	0	Muffin
28	45	11	17	17	8	1	0	0	Muffin

Step 2# Preparing The Data

Is the process of transforming and understanding raw data so that data scientists and analysts can run it through machine learning algorithms to uncover insights or make predictions;

Below we find general statistics, below we see that the highest - max sugar level was 52, the least- min number of Eggs were 0, etc..etc

```
data.describe()
```

	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Sal
count	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	50.000000	50.0000
mean	39.340000	15.920000	23.080000	9.740000	6.940000	2.680000	0.960000	0.9600
std	11.213348	8.090912	14.037617	4.898188	3.644453	3.152194	2.732719	1.9893
min	19.000000	0.000000	3.000000	3.000000	0.000000	0.000000	0.000000	0.0000
25%	34.000000	11.000000	16.000000	6.000000	5.000000	1.000000	0.000000	0.0000
50%	39.000000	17.000000	20.000000	8.000000	6.000000	1.000000	0.000000	0.0000
75%	47.750000	23.000000	26.000000	12.000000	9.000000	2.000000	1.000000	0.0000

Step 3# Spliting Data

This is a process involves splitting data to to features and outcome, in this case features are **Flour Milk Sugar Butter Egg Baking Powder Vanilla Salt** and outcome is **Type Column** Refer from the dataset in th first cell

```
# Split to X, Y
array = data.values
X = array[:, 0: 8]    # Flour Milk Sugar Butter Egg Baking Powder Vanilla Salt 1
Y = array[:, 8]  # Type
```

Step 4# Spliting Data

In this step we split data into Training Set - 70% and Testing Set - 30%. Please install sklearn. pip3 install sklearn

```
from sklearn import model_selection
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y,
                                                                    test_size=0.30,
                                                                    random_state=42)
```

Step 4a# - Model Cross Validation. KFOLD

This step shows model Cross Validation, where we check the most likely performance when Fitted to our dataset.

```
# Imports Classification Models
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC

# Create an Empty List and append all models, Give the Models an alias Nam
# i.e we called KNeighborsClassifier() alias KNC etc
models = []
models.append(('D Trees', DecisionTreeClassifier()))
models.append(('Gaussian', GaussianNB()))
models.append(('KNC', KNeighborsClassifier()))
models.append(('Random Forest', RandomForestClassifier()))
models.append(('Gradient Boosting', GradientBoostingClassifier()))
models.append(('Linear Disc', LinearDiscriminantAnalysis()))
models.append(('Support Machines', SVC(gamma = 'auto')))

# Import Cross Validation and KFOLD
```

```
from sklearn.model_selection import cross_val_score, KFold
# Create a for loop so that each model is tested in turn
for name, model in models:
     # Here we do a 10 split K-FOLD
     kfold = KFold(n_splits = 10, random_state=42, shuffle=True)

     # We get the results for each Fold
     cv_results = cross_val_score(model, X_train, Y_train, cv = kfold,
     scoring='accuracy')

     # Get the average of all Folds
     print(name, '  Results:= ', cv_results.mean())
```

```
D Trees   Results:=  1.0
Gaussian   Results:=  0.95
KNC   Results:=  0.925
Random Forest   Results:=  0.975
Gradient Boosting   Results:=  1.0
Linear Disc   Results:=  0.975
Support Machines   Results:=  0.6916666666666667
```

Above we see that most models/algorithms would score 100%. But some like Gaussian, KNC would score 95% and Support Vector Machines would score as low as 69%. Next we use the Model that performed better. I.e Gradient Boosting NB: We can also try to use Support Machines and observe how it Performs. Please note the cross validation results can slightly change when used in Training our Model, Cross validation only tries to Guide you in selecting the best model from Many.

Step 5# Choosing a Model

This is a process that can be applied both across different types of models (e.g. logistic regression, SVC, KNN, gausian etc.);

After Choosing a Model Fit the Training data in it and hide the Testing data

```
# Load classicaftion model
# we use GradientBoostingClassifier as it performed better in Step 4a(100%) - Cross Validation
from sklearn.ensemble import GradientBoostingClassifier

model = GradientBoostingClassifier()
model.fit(X_train, Y_train)
```

```
▾ GradientBoostingClassifier
GradientBoostingClassifier()
```

Step 6# Test Model For Accuracy

In this Step we test the model to check the accuracy, we use the 30% data. Please Note this data was not fitted in training the model

```
predictions = model.predict(X_test)
print('Model Predicted ', predictions)
print('Expected ', Y_test)
```

```
Model Predicted  ['Cupcake' 'Cupcake' 'Muffin' 'Scone' 'Cupcake' 'Scone' 'Muffin' 'Muffin'
 'Muffin' 'Cupcake' 'Cupcake' 'Muffin' 'Cupcake' 'Muffin' 'Muffin']
Expected  ['Cupcake' 'Cupcake' 'Muffin' 'Scone' 'Cupcake' 'Scone' 'Muffin' 'Muffin'
 'Muffin' 'Cupcake' 'Cupcake' 'Muffin' 'Cupcake' 'Muffin' 'Muffin']
```

Step 7# Get Model Accuracy

Here we check the modle accuracy

```
from sklearn.metrics import accuracy_score
print ('Accuracy ', accuracy_score(Y_test, predictions))
```

```
Accuracy  1.0
```

Step 8# use the model to check what cake you are cooking

Here we only provide the model with ingradients and it predicts the cake

```
recipe = [[19,  5,  25, 5,  0   ,10 ,0, 5   ]]
```

```
outcome = model.predict(recipe)
print('You will be cooking a ', outcome)
```

```
     You will be cooking a  ['Scone']
```

And the Model does it!, we are preparing a scone, what's the difference between a muffin and a cupcake? Scones? turns out muffins have more flour, while cupcakes have more butter and sugar.