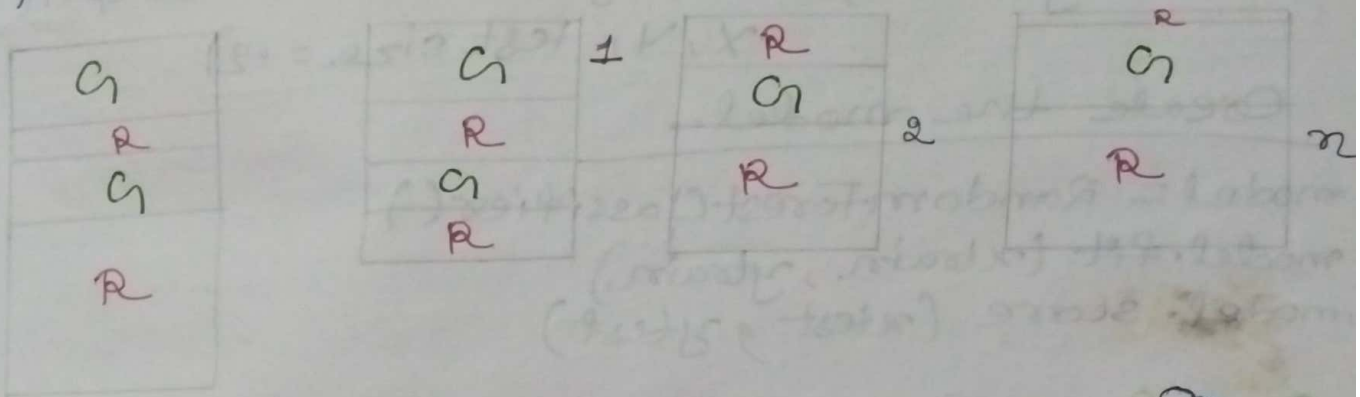
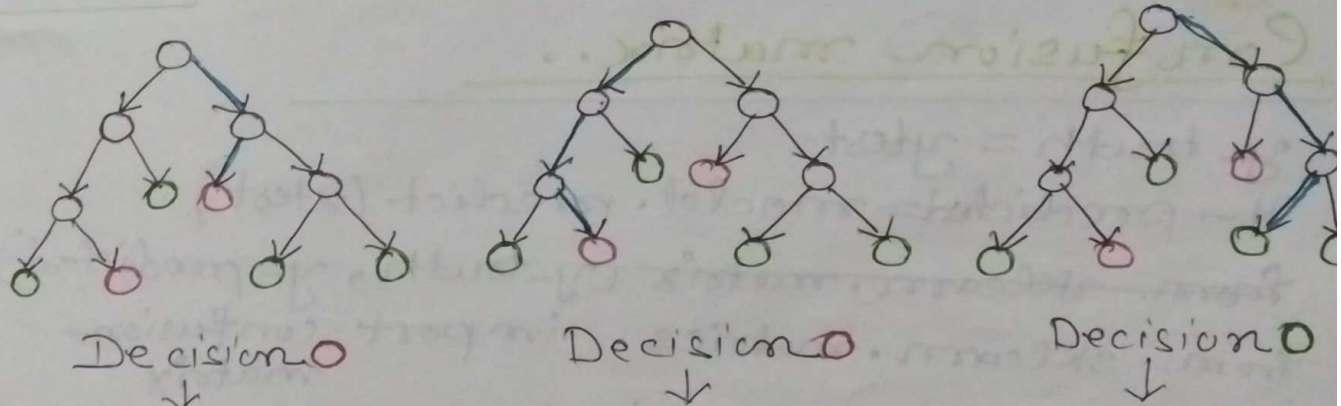
RANDOM FOREST....

Random forest algorithm is another machine learning technique.
from the name, it's forest. Forest means too many trees. And tree in machine learning is decision tree algorithm.
So its randomly creating number of decision tree from a dataset by choosing number of random datasets.
And each decision tree have a decision and you choose the majority one.
That's all about randomforest.





In notebook

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
digits= load_digits()
```

Cross check the data..

① for i in range(3):
    plt.matshow(digits.images[i])

② digits.target
③ digits.target_names

Create the data frame and split..

```
data = pd.DataFrame(digits.data)
data['target'] = digits.target
x = data.drop(['target'], axis=1)
y = data.target
xtrain, ytrain, xtest, ytest = train_test_split(
                    x, y, test_size = .2)
```

Create the model..

```
model = RandomForestClassifier()
model.fit(xtrain, ytrain)
model.score(xtest, ytest)
```

Confusion matrix..

```
y_truth = ytest
y_predicted = model.predict(xtest)
from sklearn.metrix (y_truth, y_predicted)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_truth, y_predicted)
plt.figure(figsize=(10,7))
seaborn.heatmap(cm, annot=True)
```

Assignment..

Use flower famous iris dataset from sklearn.dataset to predict flower species using random forest classifier.

① Measure prediction score using default n_estimators (10).

② Now fine tune your model by changing number of trees in your classifier and tell me what

CROSS VALIDATION..

Evaluating model performance..

We have different machine learning classifie like logistic regression, decision tree, random forest, SVM.

But when we are dealing with a problem or a dataset, how we will choose a classifier For that we should know which one is scoring better. So for that purpose we can choose cross validation.

Different ways to do validate..

Option1: Use the whole dataset for training and use the same dataset for testing.

But this is not the best method because while we are training the data the model already seen the data and we are testing on the same data.

Option2: Split the available dataset into train and test.

```
from sklearn.model_selection train_test_split.
xtrain, xtest, ytrain, ytest = train_test_spli
                                (X, Y, test_size = 0.3)
```

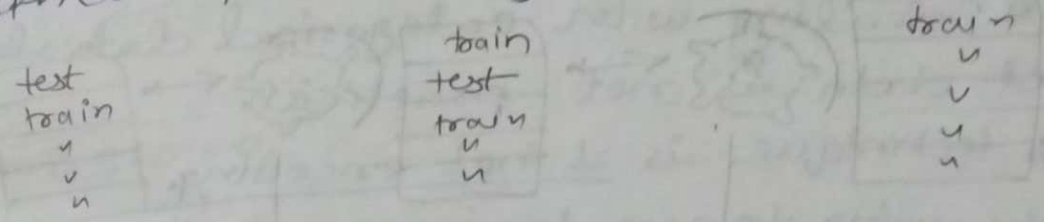
The 1st problem with this approach is the splitted data may not be uniform, for example if we are training some maths questions to our model and when we splits consider we got our train model full of trignometry questions and the test contains full of exponential questions, then, the model will fail.

The 2nd problem with this approach is, when ever you run the above code it will give different different datasets. So the score will vary. So we need to take no. of times to validate validate which classifier performs better.

Option 3

K FOLD cross validation..

In this we first devide the whole dataset into number of folds and will have that many iterations also. i.e in each iteration one fold will be test and all other folds are for train the data and for next iteration it will take another fold for testing and all others for training.



100 samples

Average

↓

Average Score

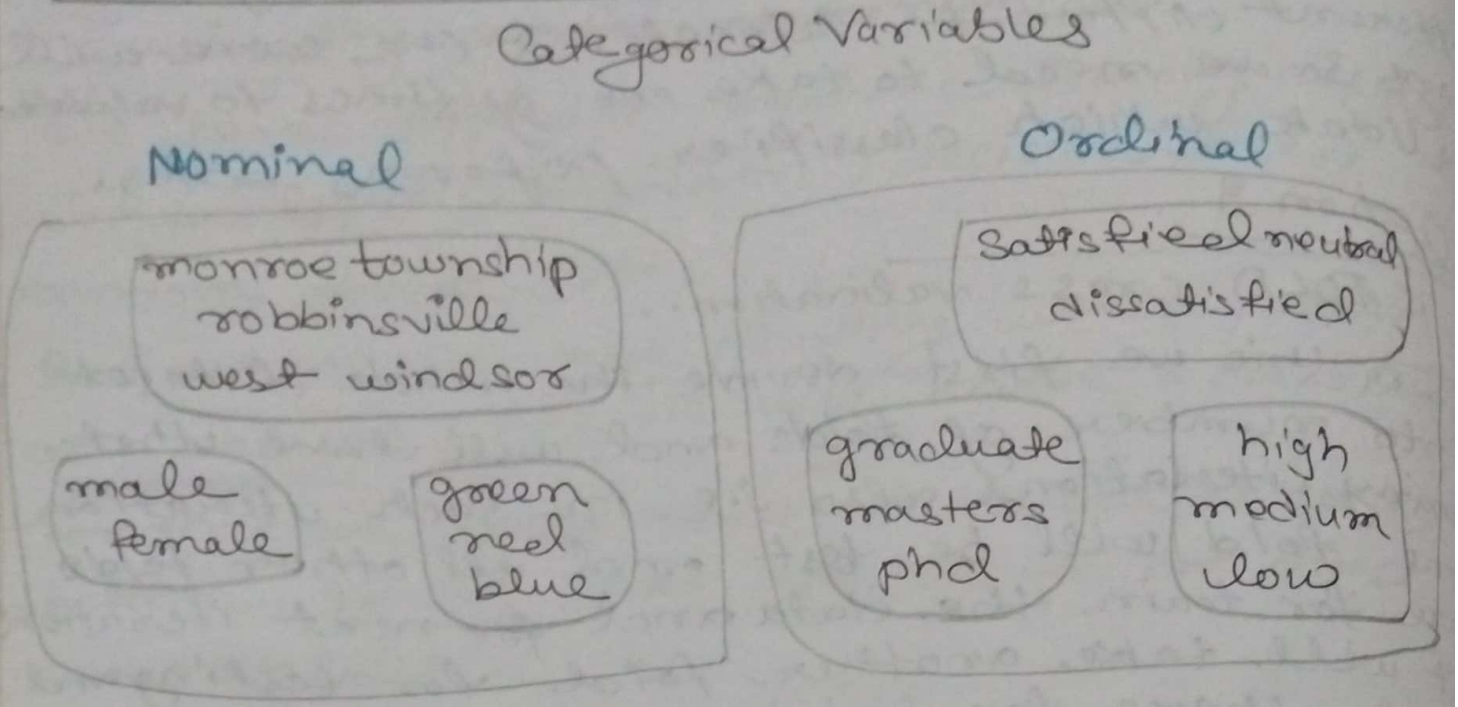
In notebook.. Kfold sample

```
from sklearn.model_selection import KFold
fold = KFold(n_splits=4)
fold
for train_index, test_index in fold.split([1,2,3,4,5,6,7,8,9,10,11,12]):
    print(train_index, test_index)
```

Lets do it with digits data set..

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import StratifiedKFold
fold = StratifiedKFold(n_split=4)
fold
```

Categorical variables..



So while dealing with categorical data label for text data is not practical.
So the technique is 1 hot encoding.
So we are going to create extra variable in our data and these datas are known as dummy variable.

Dummy variables..

town	area	price	monroe township	west windsor	robbinsville
monroe township	2600	550000	1	0	0
"	3000	6..	1	0	0
"	4000		1	0	0
"	1000		1	0	0
"	9000		1	0	0
"					
west windsor	2980		0	1	0
"	3240		0	1	0
"	6782		0	1	0
"	3214		0	1	0
robbinsville	5 346		0	0	1
"	2057		0	0	1

Create the Data

```
data = pd.read_csv ('homeprices.csv')
dummies = pd.get_dummies (data.town)
merge = pd.concat ([data, dummies],
                   axis=1)
merge = merge.drop (['town'], axis=1)
merge.
```

Train the model

```
X = merge.drop (['price'], axis=1)
Y = merge.price.
model = linear_model.LinearRegression()
model.fit (X, Y)
model.predict ([[2000, 0, 1, 0]])
```

score

```
model.score (X, Y)
```

Using 1 hot encoder.

Creating X and Y

```
df = data
df
from sklearn.preprocessing import OneHotEncoder

Y = df.price
from sklearn.compose import ColumnTransform

ts = ColumnTransformer (
    transformers = [
      ("abc", OneHotEncoder(), [1])
    ],
    remainder = 'passthrough')
    X = ts.fit_transform (X)
```

Train And Predict

```
model = linear_model.LinearRegression()
model.fit (X, Y)
model.predict ([[1, 0, 0, 2000]])
```

Get the scores method..

```
def get_score_from_model(model, xtrain, xtest,
                         ytrain, ytest):
    model.fit(xtrain, ytrain)
    return model.score(xtest, ytest)
```

Find the scores..

```
logistic_scores = []
rf_scores = []
SVC_scores = []
for train_index, test_index in fold.split(digits.
                data, digits.target):
    xtrain, xtest, ytrain, ytest = digits.data[train_
            index], digits.data[test_index], \
            digits.target[train_index], digits.
                target[test_index]
    logistic_scores.append(get_score_from_model
            (LogisticRegression(), xtrain,
            xtest, ytrain, ytest))
    rf_scores.append(get_score_from_model
            (RandomForestClassifier(), xtrain,
            xtest, ytrain, ytest))
    SVC_score.append(get_score_from_model
            (SVC(), xtrain, xtest, ytrain, ytest))
print(logistic_scores)
print(rf_scores)
print(SVC_scores)
```

You can take the average from the list and find which classifier performs better..

In short way..

```
from sklearn.model_selection import cross_val_score
l_score = cross_val_score(LogisticRegression(), X = digits.
          data, y = digits.target)
svc_scores = cross_val_score(SVC(), X = digits.data,
             y = digits.target)
rf_scores = cross_val_score(RandomForestClassifier
            (), X = digits.data, y = digits.target)
print(l_scores)
print(svc_scores)
print(rf_scores)
```

Exercise..

1) Using cross validation find which classifier is better to predict iris dataset.