In [1]:
```
#-[p(yes)*log(p(yes))+p(no)*log(p(no))]
import numpy as np
-((3/6)* np.log(3/6)+(3/6)*np.log(3/6))
```

Out[1]:
```
0.6931471805599453
```

In [2]:
```
import pandas as pd
```

In [3]:
```
data = pd.read_csv("diabetes.csv")
data.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [4]:
```
data.shape
```

Out[4]:
```
(768, 9)
```

In [5]:
```
data.columns
```

Out[5]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [6]:
```
# preprocessing
data.isnull().sum()
```

Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```
1 data.isnull().count()
```

Out[7]:
```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [8]:
```
1 data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [9]:
```
1 data.Outcome.value_counts()
```

Out[9]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [10]:
```
1 # spliting features and target
2 x = data.drop("Outcome",axis=1)
```

In [11]:
```
1 y = data["Outcome"]
```

In [12]:
```
1 # spliting the data for training and testing
2
3 from sklearn.model_selection import train_test_split
```

In [95]:
```
1 x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.33,random_s
```

In [96]:
```
1 # select the model
2 from sklearn.ensemble import RandomForestClassifier
```

In [97]:
```
rfc = RandomForestClassifier()
```

In [98]:
```
rfc.fit(x_train,y_train)
```

```
C:\Users\Alekhya\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: Fu
tureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[98]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [99]:
```
# predict
pred = rfc.predict(x_test)
```

In [100]:
```
pred
```

Out[100]:
```
array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0], dtype=int64)
```

In [101]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_
```

In [102]:
```
confusion_matrix(y_test,pred)
```

Out[102]:
```
array([[145,  31],
       [ 37,  41]], dtype=int64)
```

In [103]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.82      0.81       176
           1       0.57      0.53      0.55        78

   micro avg       0.73      0.73      0.73       254
   macro avg       0.68      0.67      0.68       254
weighted avg       0.73      0.73      0.73       254
```

In [104]:
```
accuracy_score(y_test,pred)
```

Out[104]:
```
0.7322834645669292
```

In [105]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [106]:
```
help(RandomizedSearchCV)
```

```
|      See :ref:`multimetric_grid_search` for an example.
|
|      If None, the estimator's default scorer (if available) is used.
|
|  fit_params : dict, optional
|      Parameters to pass to the fit method.
|
|      .. deprecated:: 0.19
|         ``fit_params`` as a constructor argument was deprecated in version
|         0.19 and will be removed in version 0.21. Pass fit parameters to
|         the ``fit`` method instead.
|
|  n_jobs : int or None, optional (default=None)
|      Number of jobs to run in parallel.
|      ``None`` means 1 unless in a :obj:`joblib.parallel_backend` context.
|      ``-1`` means using all processors. See :term:`Glossary <n_jobs>`
|      for more details.
|
|  pre_dispatch : int, or string, optional
|      Controls the number of jobs that get dispatched during parallel
```

In [109]:
```
# number of tress in random forest
n_estimators = [int(i) for i in np.linspace(200,2000,10)]
```

In [110]:
```
n_estimators
```

Out[110]:
```
[200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000]
```

In [111]:
```
# number of features at every spliting
max_features = ["auto","sqrt"]
```

In [112]:
```
# max_depth
max_depth = [int(x) for x in np.linspace(100,500,11)]
max_depth
```

Out[112]:
```
[100, 140, 180, 220, 260, 300, 340, 380, 420, 460, 500]
```

In [113]:
```
#param_distributions
param = {"n_estimators":n_estimators,"max_features":max_features,"max_depth"
param
```

Out[113]:
```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],
 'max_features': ['auto', 'sqrt'],
 'max_depth': [100, 140, 180, 220, 260, 300, 340, 380, 420, 460, 500]}
```

In [114]:
```
## searching parameters
rfc_random = RandomizedSearchCV(estimator=rfc,param_distributions=param,
                    n_iter=100,cv=3,verbose=3,random_state=45,n_jobs=-1)
```

In [115]:
```
rfc_random.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  24 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 120 tasks      | elapsed:  4.2min
[Parallel(n_jobs=-1)]: Done 280 tasks      | elapsed:  9.4min
[Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed:  9.9min finished
```

Out[115]:
```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
          estimator=RandomForestClassifier(bootstrap=True, class_weight=None, c
riterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False),
          fit_params=None, iid='warn', n_iter=100, n_jobs=-1,
          param_distributions={'n_estimators': [200, 400, 600, 800, 1000, 1200,
1400, 1600, 1800, 2000], 'max_features': ['auto', 'sqrt'], 'max_depth': [100, 1
40, 180, 220, 260, 300, 340, 380, 420, 460, 500]},
          pre_dispatch='2*n_jobs', random_state=45, refit=True,
          return_train_score='warn', scoring=None, verbose=3)
```

In [116]:
```
rfc_random.best_params_
```

Out[116]:
```
{'n_estimators': 200, 'max_features': 'sqrt', 'max_depth': 180}
```

In [119]:
```
rfc1 = RandomForestClassifier(n_estimators=200,max_features='sqrt',max_depth
```

In [120]:
```
rfc1.fit(x_train,y_train)
```

Out[120]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=180, max_features='sqrt', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=200, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [121]:
```
rfc1_pred = rfc1.predict(x_test)
```

In [122]:
```
rfc1_pred
```

Out[122]:
```
array([0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1], dtype=int64)
```

In [123]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_
```

In [124]:
```
accuracy_score(y_test,rfc1_pred)
```

Out[124]:
```
0.7559055118110236
```

In [126]:
```
confusion_matrix(y_test,rfc1_pred)
```

Out[126]:
```
array([[148,  28],
       [ 34,  44]], dtype=int64)
```

In [129]:
```
from sklearn.model_selection import cross_val_score
```

In [137]:
```
r = cross_val_score(rfc1,x,y,cv=15,scoring="roc_auc")
r
```

Out[137]:
```
array([0.70588235, 0.85784314, 0.86111111, 0.84477124, 0.72712418,
       0.75673401, 0.75589226, 0.93855219, 0.80387205, 0.84259259,
       0.87289562, 0.93771044, 0.83922559, 0.78253119, 0.89483066])
```

In [138]:
```
r.mean()
```

Out[138]:
```
0.8281045751633989
```

- do regression dataset for random forest regressor

datasetlink (https://drive.google.com/file/d/1ok_BeaV0VWXCBJ6HViEm3GvT9T2mgJD1/view)

In [ ]:
```
1
```