In [68]:
```
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
```

In [69]:
```
import pandas as pd
```

Till just we import the important libraries we need to implement the python code for gender identification

In [70]:
```
train_data = pd.read_csv(r"F:\Semester 6\Machine Learning\train.csv")
```

In [71]:
```
test_data = pd.read_csv(r"F:\Semester 6\Machine Learning\test.csv")
```

Now we just read the csv files for test and train data

In [72]:
```
train_data
```

Out[72]:

| | height | weight | hair | beard | scarf | gender |
|---|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | Bald | Yes | No | Male |
| **1** | 170.0000 | 120 | Long | No | No | Female |
| **2** | 178.5000 | 200 | Short | No | No | Male |
| **3** | 163.4000 | 110 | Medium | No | Yes | Female |
| **4** | 175.2222 | 220 | Short | Yes | No | Male |
| **5** | 165.0000 | 150 | Medium | No | Yes | Female |

In [73]:
```
test_data
```

Out[73]:

| | height | weight | hair | beard | scarf | gender |
|---|---|---|---|---|---|---|
| **0** | 179.1 | 185 | Long | Yes | No | Male |
| **1** | 160.5 | 130 | Short | No | No | Female |
| **2** | 177.8 | 160 | Bald | No | No | Male |
| **3** | 161.1 | 100 | Medium | No | No | Female |

# Encoding the categorical values to continous values by using label encoder()

In [25]:
```
train = train_data
```

In [26]:
```
test = test_data
```

In [74]:
```
le = LabelEncoder()
```

In [76]:
```
categorical_feature_mask = train_data.dtypes==object
```

In [77]:
```
categorical_cols = train_data.columns[categorical_feature_mask].tolist()
```

In [78]:
```
train_data[categorical_cols] = train_data[categorical_cols].apply(lambda col:
```

In [80]:
```
train_data
```

Out[80]:

| | height | weight | hair | beard | scarf | gender |
|---|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | 0 | 1 | 0 | 1 |
| **1** | 170.0000 | 120 | 1 | 0 | 0 | 0 |
| **2** | 178.5000 | 200 | 3 | 0 | 0 | 1 |
| **3** | 163.4000 | 110 | 2 | 0 | 1 | 0 |
| **4** | 175.2222 | 220 | 3 | 1 | 0 | 1 |
| **5** | 165.0000 | 150 | 2 | 0 | 1 | 0 |

In [81]:
```
categorical_feature_mask = test_data.dtypes==object
```

In [82]:
```
categorical_cols = test_data.columns[categorical_feature_mask].tolist()
```

In [83]:
```
test_data[categorical_cols] = test_data[categorical_cols].apply(lambda col: l
```

In [84]:
```
test_data
```

Out[84]:

| | height | weight | hair | beard | scarf | gender |
|---|---|---|---|---|---|---|
| **0** | 179.1 | 185 | 1 | 1 | 0 | 1 |
| **1** | 160.5 | 130 | 3 | 0 | 0 | 0 |
| **2** | 177.8 | 160 | 0 | 0 | 0 | 1 |
| **3** | 161.1 | 100 | 2 | 0 | 0 | 0 |

Split the train data into input and output so we train our models

In [85]:
```
X_train = train_data.iloc[: , 0:5]
```

In [86]:
```
X_train
```

Out[86]:

| | height | weight | hair | beard | scarf |
|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | 0 | 1 | 0 |
| **1** | 170.0000 | 120 | 1 | 0 | 0 |
| **2** | 178.5000 | 200 | 3 | 0 | 0 |
| **3** | 163.4000 | 110 | 2 | 0 | 1 |
| **4** | 175.2222 | 220 | 3 | 1 | 0 |
| **5** | 165.0000 | 150 | 2 | 0 | 1 |

In [91]:
```
Y_train = train_data.iloc[: , 5:]
```

In [92]:
```
Y_train
```

Out[92]:

| | gender |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |
| **4** | 1 |
| **5** | 0 |

Now doing the same with test data split into input and output

In [93]:
```
X_test = test_data.iloc[: , 0:5]
```

In [94]:
```
X_test
```

Out[94]:

| | height | weight | hair | beard | scarf |
|---|---|---|---|---|---|
| **0** | 179.1 | 185 | 1 | 1 | 0 |
| **1** | 160.5 | 130 | 3 | 0 | 0 |
| **2** | 177.8 | 160 | 0 | 0 | 0 |
| **3** | 161.1 | 100 | 2 | 0 | 0 |

In [95]:
```
Y_test = test_data.iloc[: , 5:]
```

In [96]:
```
Y_test
```

Out[96]:

| | gender |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |

# Phase 1 : TRAINING

***In first phase we need to train our model with machine learning algorithms***

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

In [98]:
```
X_train
```

Out[98]:

| | height | weight | hair | beard | scarf |
|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | 0 | 1 | 0 |
| **1** | 170.0000 | 120 | 1 | 0 | 0 |
| **2** | 178.5000 | 200 | 3 | 0 | 0 |
| **3** | 163.4000 | 110 | 2 | 0 | 1 |
| **4** | 175.2222 | 220 | 3 | 1 | 0 |
| **5** | 165.0000 | 150 | 2 | 0 | 1 |

In [99]:
```
rfc_clf.fit(X_train,Y_train)
```

```
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\ensemble\forest.p
y:246: FutureWarning: 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)
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: D
ataConversionWarning: A column-vector y was passed when a 1d array was expe
cted. Please change the shape of y to (n_samples,), for example using ravel
().
  """Entry point for launching an IPython kernel.
```

Out[99]:
```
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 [100]:
```
l_clf = LogisticRegression()
```

In [101]:
```
l_clf.fit(X_train,Y_train)
```

```
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\linear_model\logis
tic.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:761: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
```

Out[101]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [102]:
```
s_clf = SVC()
```

In [103]:
```
s_clf.fit(X_train,Y_train)
```

```
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:761: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: F
utureWarning: The default value of gamma will change from 'auto' to 'scale'
in version 0.22 to account better for unscaled features. Set gamma explicit
ly to 'auto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```

Out[103]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='rbf', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
```

In [104]:
```
ber_clf = BernoulliNB()
```

In [105]:
```
ber_clf.fit(X_train , Y_train)
```

```
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:761: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
```

Out[105]:
```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

# Phase 2 : TESTING

In [123]:
```
rfc_prediction = rfc_clf.predict(X_test)
print (rfc_prediction)
```

```
[1 0 0 0]
```

In [107]:
```
l_prediction = l_clf.predict(X_test)
print(l_prediction)
```

```
[1 0 0 0]
```

In [108]:
```
s_prediction = s_clf.predict(X_test)
print (s_prediction)
```

```
[0 0 0 0]
```

In [109]:
```
ber_prediction = ber_clf.predict(X_test)
print (ber_prediction)
```

```
[1 0 1 0]
```

### Checking the accuracy of best model

In [110]:
```
ber_acc = accuracy_score(ber_prediction,Y_test)
rfc_acc = accuracy_score(rfc_prediction,Y_test)
l_acc = accuracy_score(l_prediction,Y_test)
s_acc = accuracy_score(s_prediction,Y_test)
```

In [111]:
```
classifiers = ['BernoulliNB', 'Random Forest', 'Logistic Regression' , 'SVC']
accuracy = np.array([ber_acc, rfc_acc, l_acc, s_acc])
max_acc = np.argmax(accuracy)
print(classifiers[max_acc] + ' is the best classifier for this problem')
```

```
BernoulliNB is the best classifier for this problem
```

# Phase 3 : APPLICATION

### *Combining Training and testing data*

In [112]:
```
data = pd.concat([train,test],axis=0,join="outer")
```

In [113]:
```
data
```

Out[113]:

| | height | weight | hair | beard | scarf | gender |
|---|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | 2 | 0 | 1 | 1 |
| **1** | 170.0000 | 120 | 0 | 1 | 1 | 0 |
| **2** | 178.5000 | 200 | 1 | 1 | 1 | 1 |
| **3** | 163.4000 | 110 | 3 | 1 | 0 | 0 |
| **4** | 175.2222 | 220 | 1 | 0 | 1 | 1 |
| **5** | 165.0000 | 150 | 3 | 1 | 0 | 0 |
| **0** | 179.1000 | 185 | 0 | 0 | 1 | 1 |
| **1** | 160.5000 | 130 | 1 | 1 | 1 | 0 |
| **2** | 177.8000 | 160 | 2 | 1 | 1 | 1 |
| **3** | 161.1000 | 100 | 3 | 1 | 1 | 0 |

## Now train all data with the best model that is Random Forest

In [114]:
```
X_data = data.iloc[: , 0:5]
```

In [115]:
```
X_data
```

Out[115]:

| | height | weight | hair | beard | scarf |
|---|---|---|---|---|---|
| **0** | 180.3000 | 196 | 2 | 0 | 1 |
| **1** | 170.0000 | 120 | 0 | 1 | 1 |
| **2** | 178.5000 | 200 | 1 | 1 | 1 |
| **3** | 163.4000 | 110 | 3 | 1 | 0 |
| **4** | 175.2222 | 220 | 1 | 0 | 1 |
| **5** | 165.0000 | 150 | 3 | 1 | 0 |
| **0** | 179.1000 | 185 | 0 | 0 | 1 |
| **1** | 160.5000 | 130 | 1 | 1 | 1 |
| **2** | 177.8000 | 160 | 2 | 1 | 1 |
| **3** | 161.1000 | 100 | 3 | 1 | 1 |

In [116]:
```
Y_data = data.iloc[: , 5:]
```

In [117]:
```
Y_data
```

Out[117]:

| | gender |
|---|---|
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |
| **4** | 1 |
| **5** | 0 |
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |

In [118]:
```
ber_clf.fit(X_data,Y_data)
```

```
C:\Users\Mahad Akbar\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:761: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, warn=True)
```

Out[118]:
```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

# Data is trained

### Now getting input from users for unseen examples

In [163]:
```
height = input("Enter the height = ")
```

```
Enter the height = 160
```

In [164]:
```
weight = input("Enter weight  = ")
```

```
Enter weight  = 150
```

In [165]:
```
hair = input("Enter the hair size (Long / short / medium / bald) = " )
```

```
Enter the hair size (Long / short / medium / bald) = Short
```

In [166]:
```
beard = input("Enter the beard (Yes / NO) = ")
```

```
Enter the beard (Yes / NO) = No
```

In [167]:
```
scarf = input("Enter the scarf (Yes / No) = ")
```

```
Enter the scarf (Yes / No) = Yes
```

In [168]:
```
unseen_data =  { 'Height' : [height] , 'Weight' : [weight] , 'Hair' : [hair]
```

In [169]:
```
df = pd.DataFrame(unseen_data)
```

In [170]:
```
df
```

Out[170]:

| | Height | Weight | Hair | Beard | Scarf |
|---|---|---|---|---|---|
| **0** | 160 | 150 | Short | No | Yes |

In [171]:
```
status_hair = {"Bald":0 ,"Long":1 , "Medium" : 2 , "Short" : 3}
df["Hair"] = df.Hair.map(status_hair)
```

In [172]:
```
status_beard = {"Yes":1 ,"No":0}
df["Beard"] = df.Beard.map(status_beard)
```

In [173]:
```
status_scarf = {"Yes":1 ,"No":0}
df["Scarf"] = df.Scarf.map(status_scarf)
```

In [174]:
```
df
```

Out[174]:

| | Height | Weight | Hair | Beard | Scarf |
|---|---|---|---|---|---|
| **0** | 160 | 150 | 3 | 0 | 1 |

In [175]:
```
ber_prediction = ber_clf.predict(df)
```

In [176]:
```
if ber_prediction == 1:
    print ("male")
else :
    print ("female")
```

```
male
```

# Our prediction for unseen data is "Male"