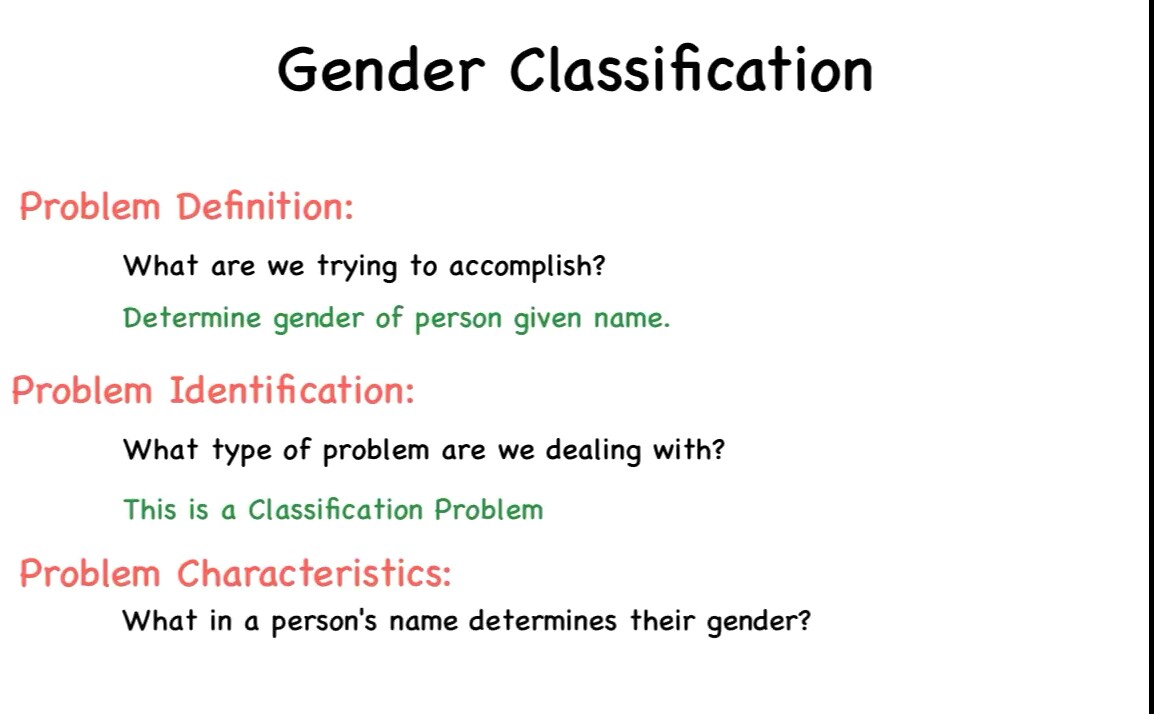
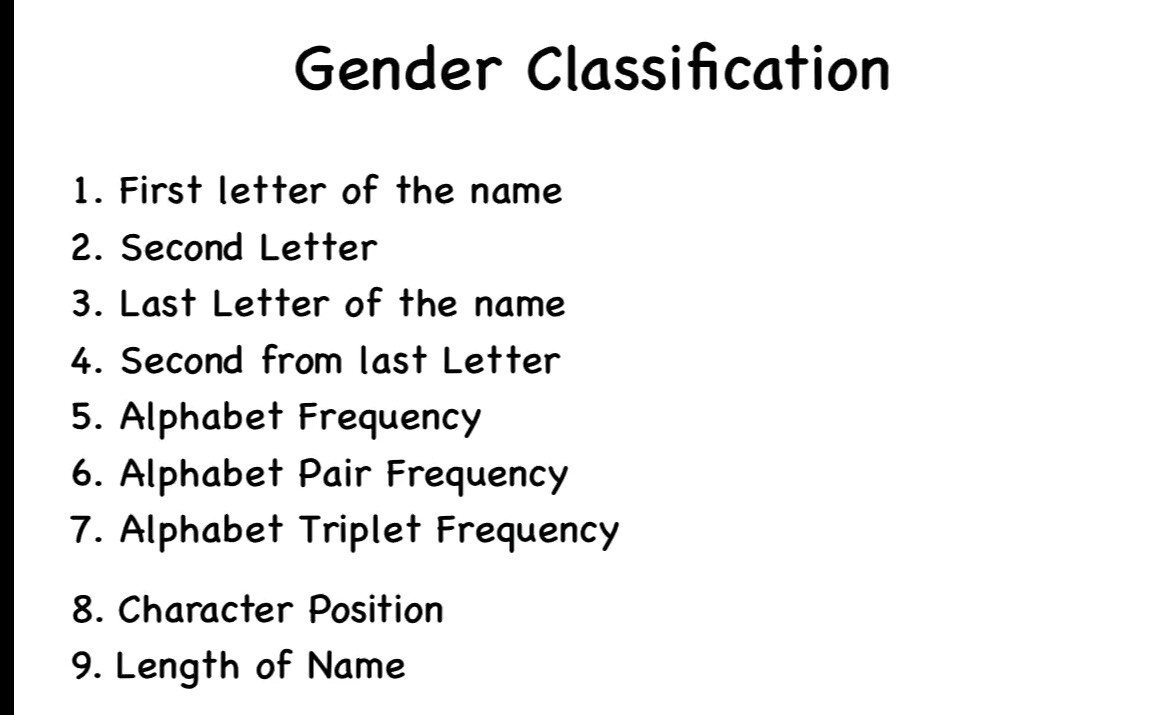
**Random Forest Classification**

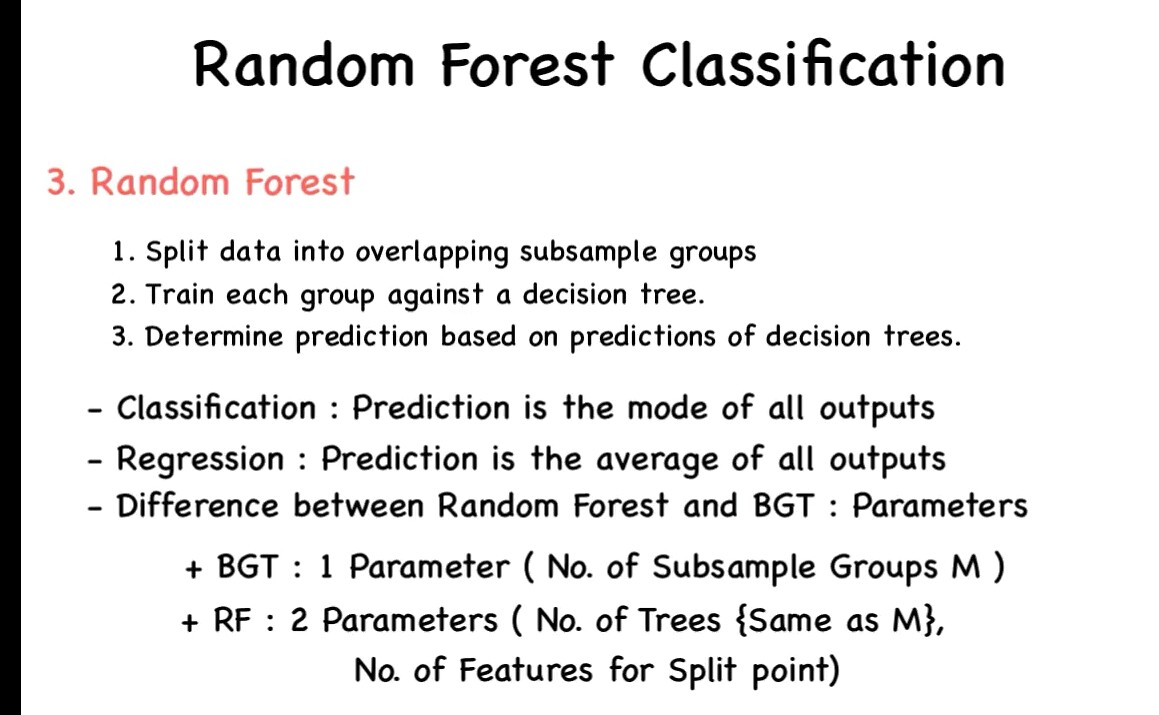
**Short note about Random Forest Classification**:

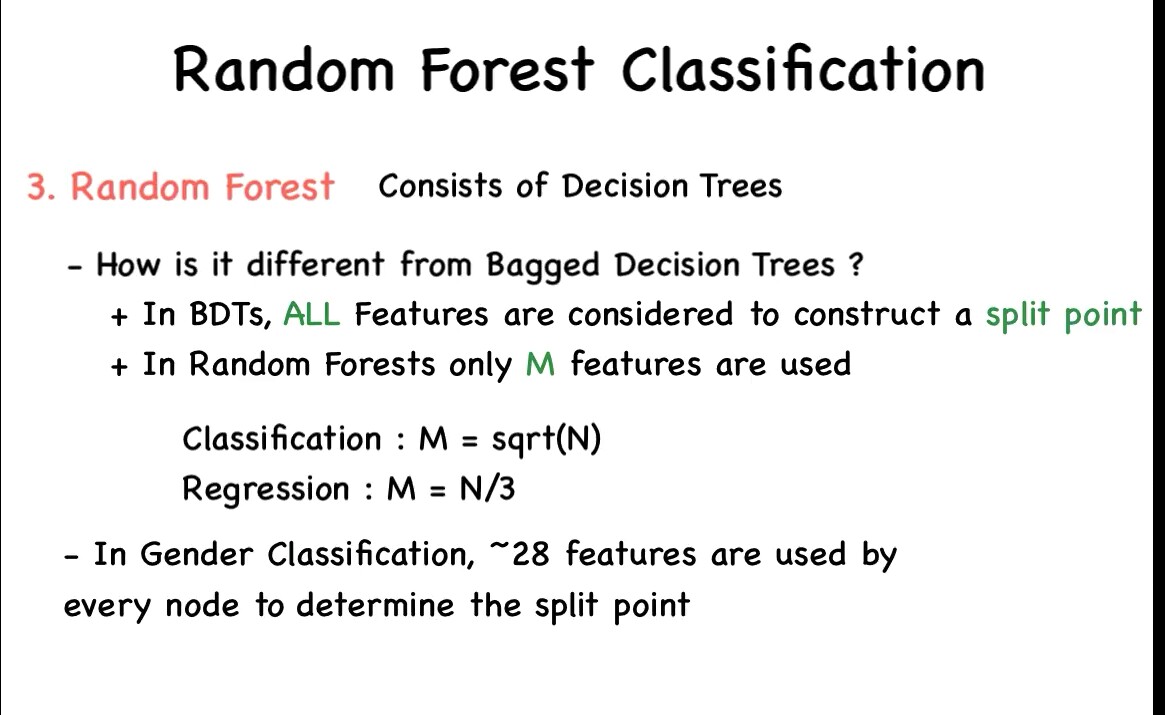
Random Forest is a statistical algorithm that is used to cluster points of data in functional groups.

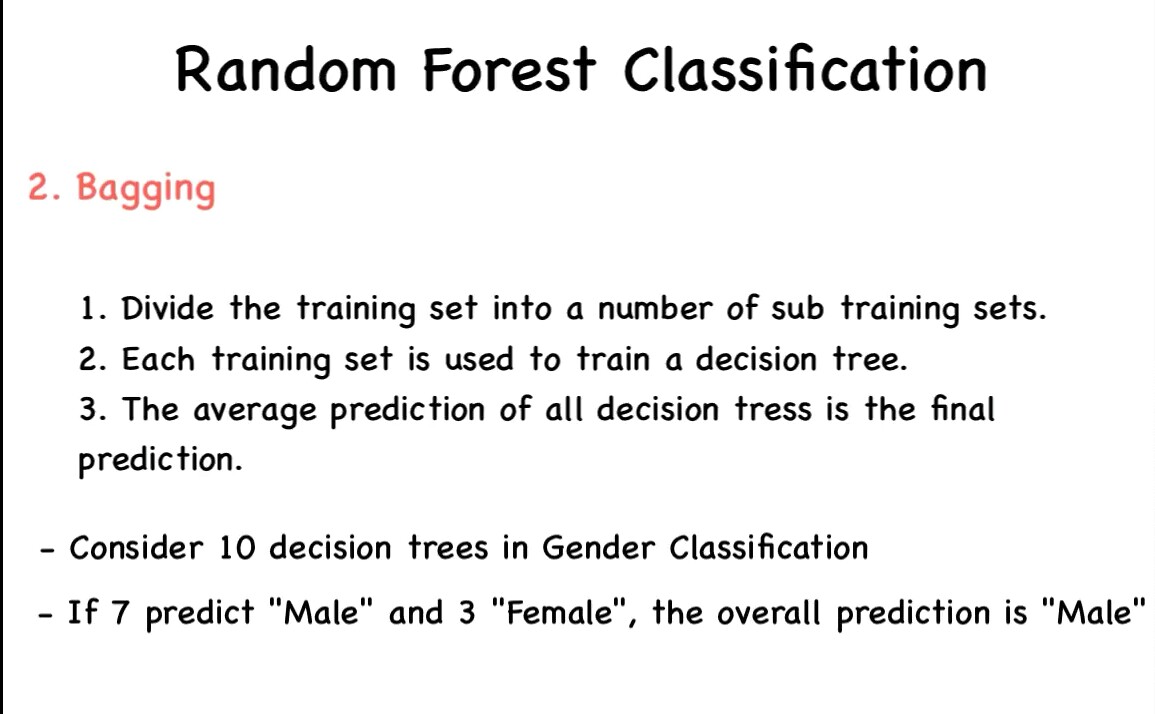
When the dataset is large and/or there are many variables it becomes difficult to cluster the data because not all variables can be taken into account.

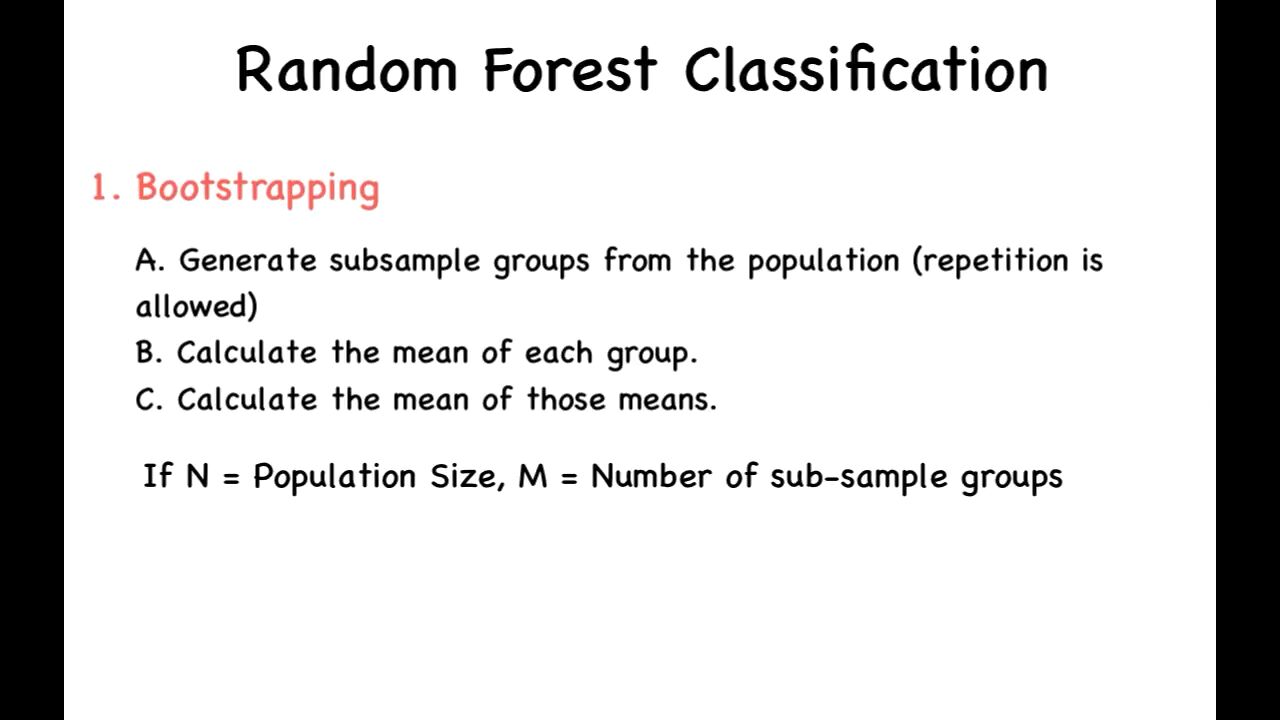


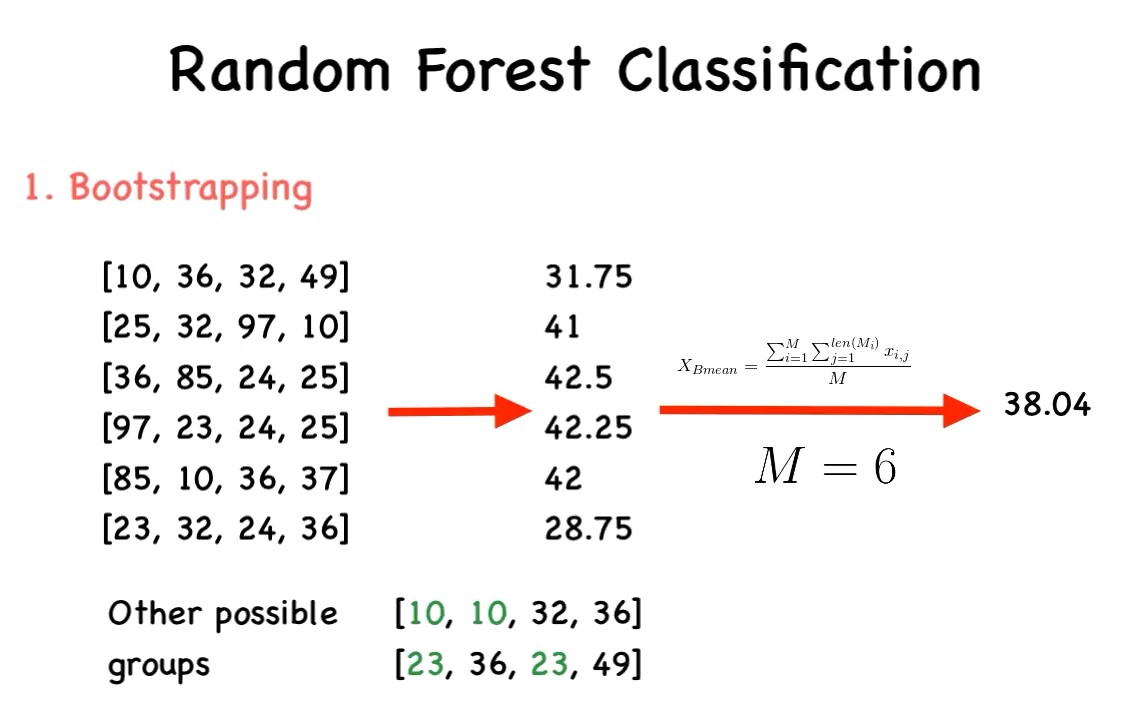


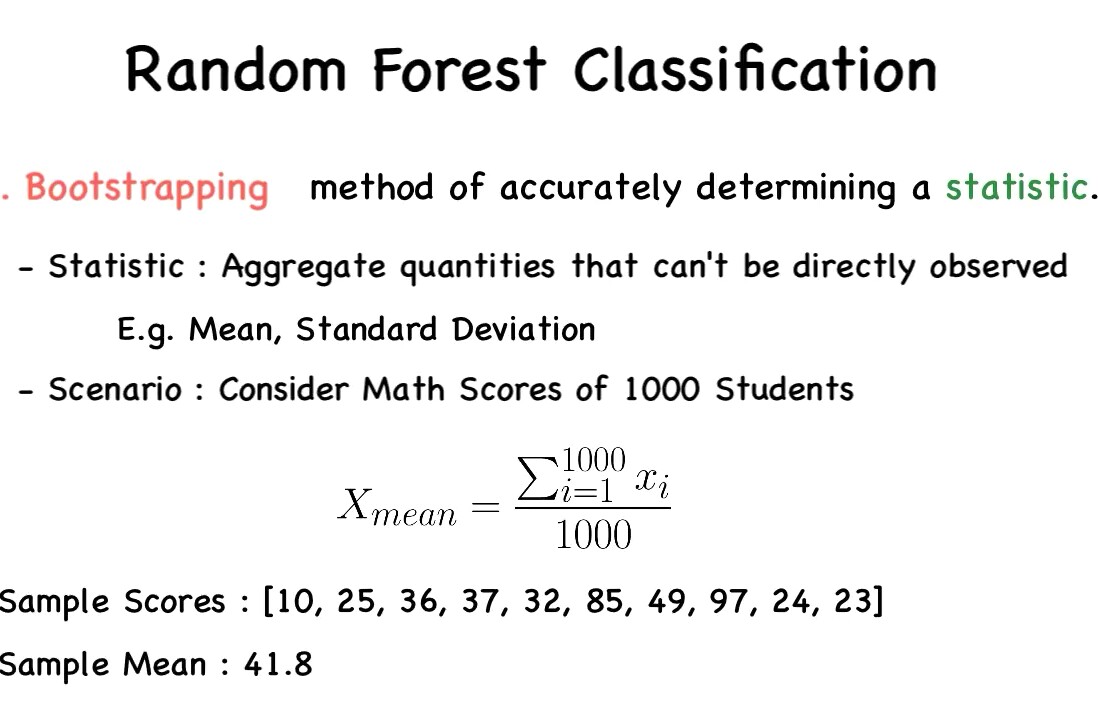


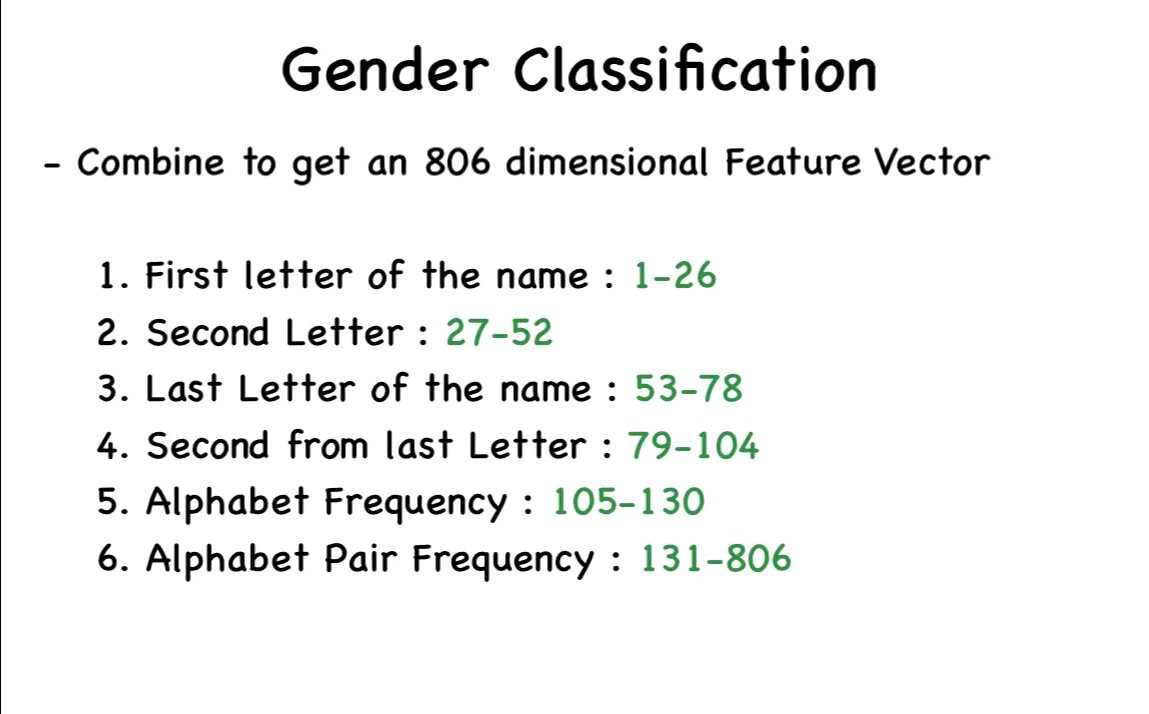


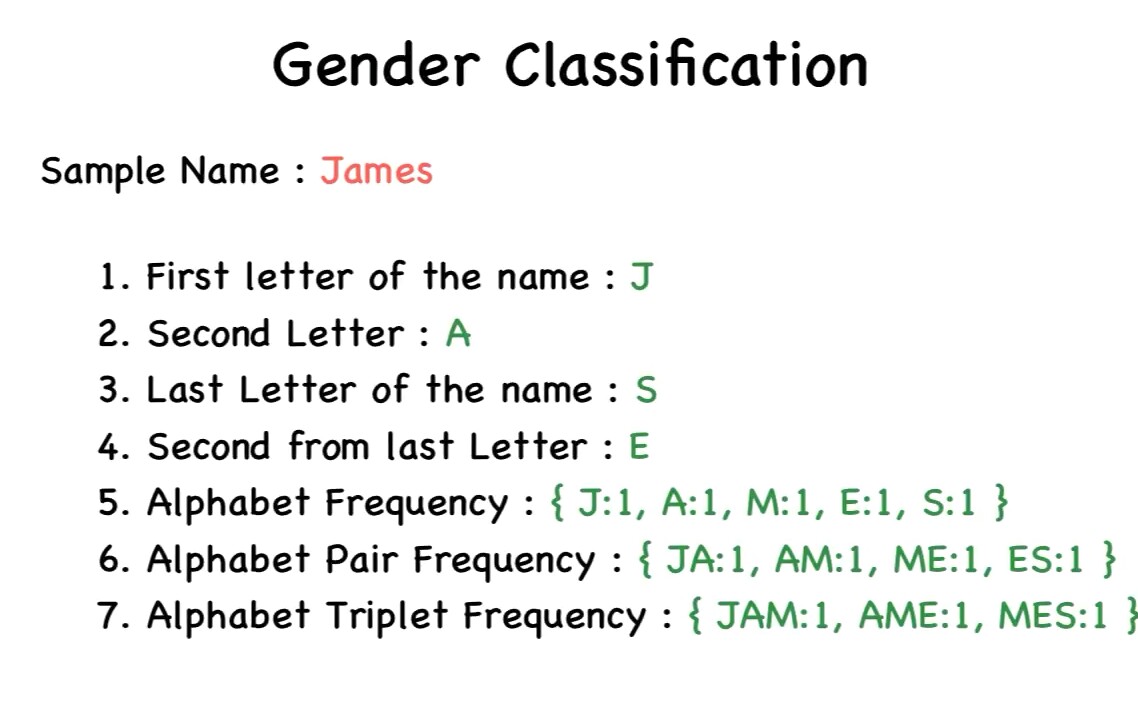


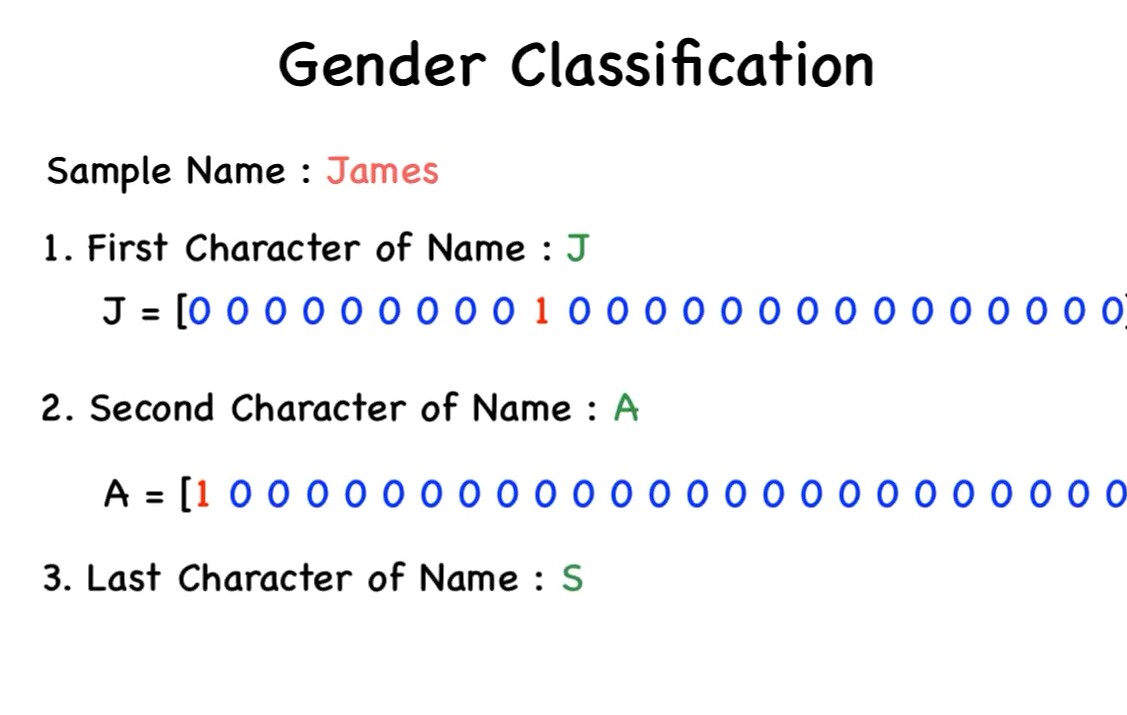


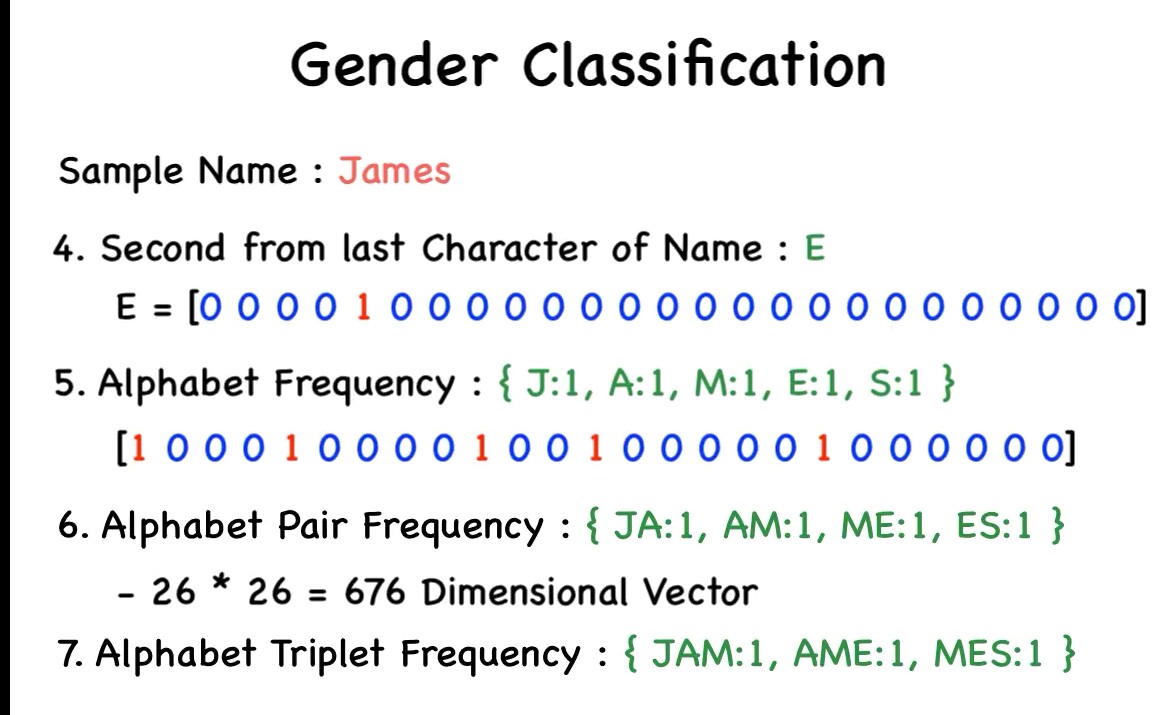




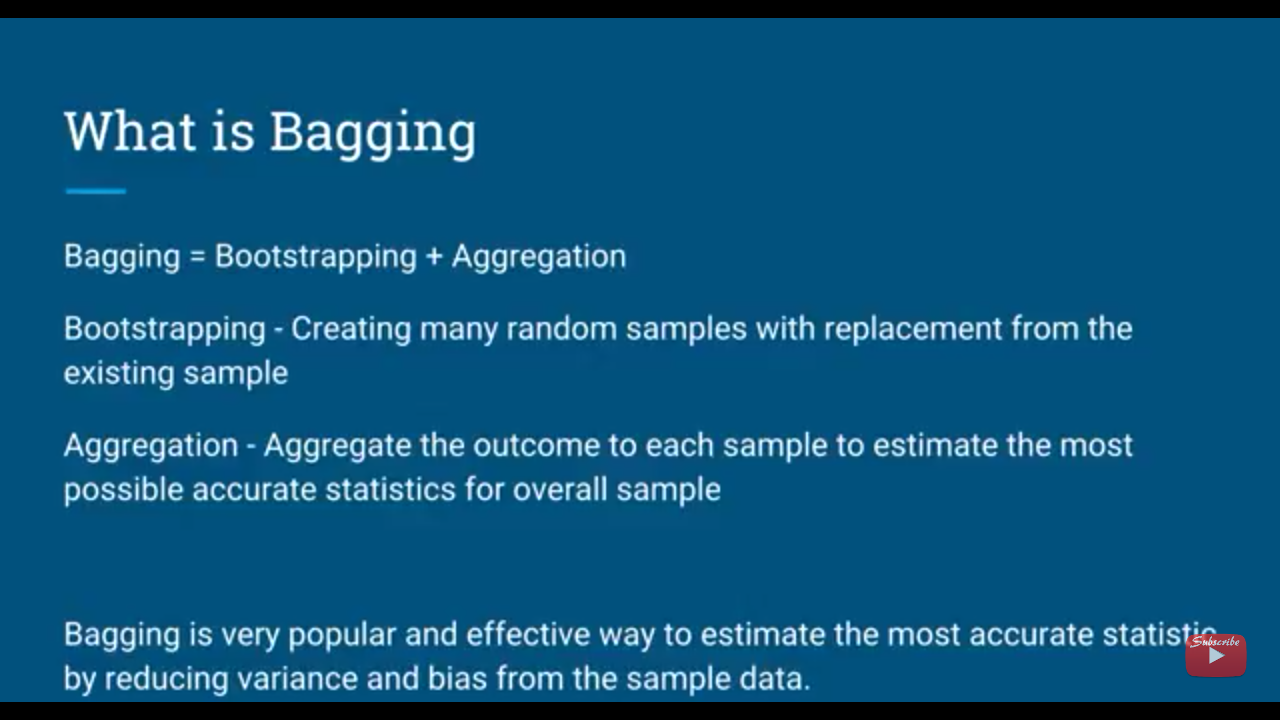


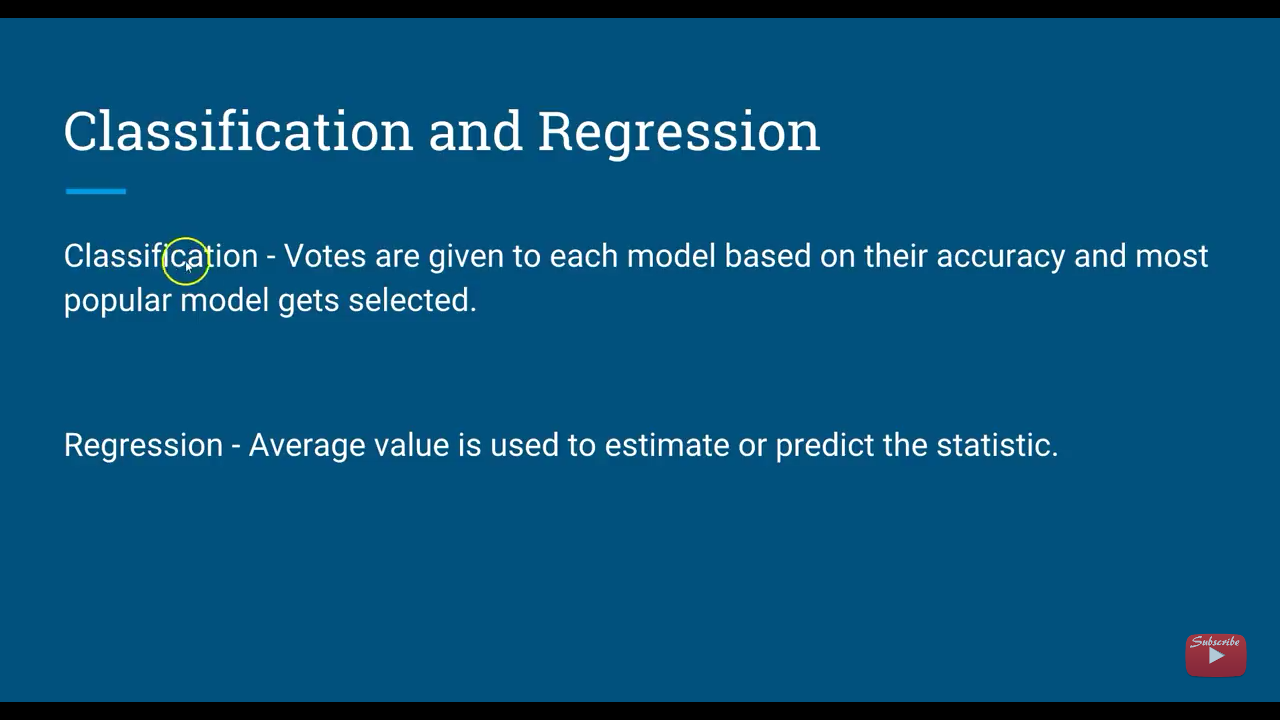






**Bagging-**





**Use Cases**

 To find the Industrial growth based on decades

Elgibility to apply for abroad

To find the capability of clearing IIT based on Marks obtained.

Elgibility to apply for credit card

Chances of getting selected in Interview

Chances of apply use cases without error.

To mention the Population Growth.

To check the Credit Risk in banks.

Finding the transactions happened in ATM per day.

Finding the changes in Pollution .

Python

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

## Importing the dataset

dataset = pd.read\_csv('C:\\Users\\Rama\\Desktop\\Social\_Network\_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values

#

# Splitting the dataset into the Training set and Test set

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25)

#

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

#

#fitting classifier to training set

from sklearn.ensemble import RandomForestClassifier

classifier=RandomForestClassifier()

classifier=RandomForestClassifier(criterion='entropy',random\_state=0)

classifier.fit=(y\_test,y\_pred)

#

#

# Fitting K-NN Regression to the Training set

from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

classifier.fit(X\_train, y\_train)

#

# Predicting the Test set results

#from sklearn.ensemble import RandomForestClassifier

y\_pred = classifier.predict(X\_test)

#

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

#

# Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title(‘RandomForestClassifier (Training set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

#

# Visualising the Test set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title(‘RandomForestClassifier (Test set)')

plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()