SURYA GROUP OF INSTITUTIONS

NAAN MUDHALVAN

IBM-ARTIFICIAL INTELLIGENCE

THARUN D

422221104042

Market Basket Insights

TEAM : 10

Market Basket Insights

Important note:

This notebook has become very large, and I could not add more information to it, so I will divide it into a number of notebooks. So you can understand the content And be an excellent reference for you

Table of Content:

Random Forest

Decision Tree

Extra Tree

Gradient Boosting

Light GBM

XGBoost

Catboost

Stochastic Gradient Descent

Lasso

Kernel Ridge Regression

Bayesian Ridge

Elastic Net Regression

LDA

K-Means Algorithm

CNN

LSTM

PCA

Apriori

Prophet

ARIMA

Machine Learning:

Linear Regression:

It is a basic and commonly used type of predictive analysis. These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. Y = a + bX where

Y – Dependent Variable

a – intercept - Bise

X – Independent variable

b – Slope -Weights

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Example: University GPA' = (0.675)(High School GPA) + 1.097

```
import numpy as np

import pandas as pd

from matplotlib import pyplot as plt
```

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

train = pd.read_csv("../input/random-linear-regression/train.csv")

test = pd.read_csv("../input/random-linear-regression/test.csv")

train = train.dropna()

test = test.dropna()

train.head()
```

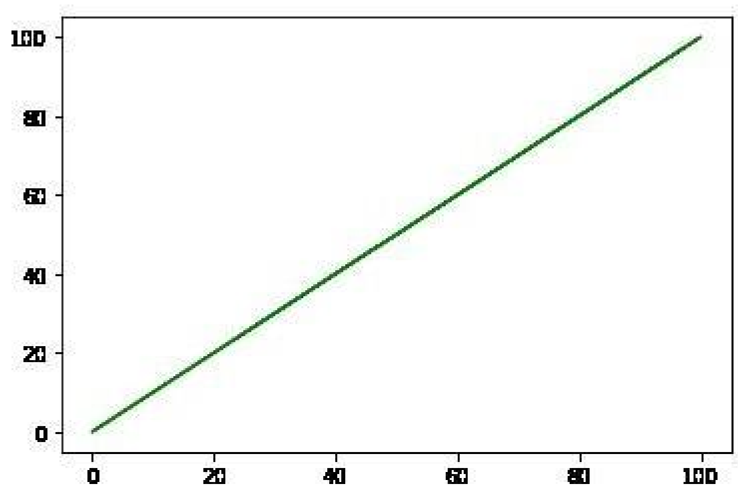
OUTPUT :

	x	y
0	24.0	21.549452
1	50.0	47.464463
2	15.0	17.218656
3	38.0	36.586398
4	87.0	87.288984

Model with plots and accuracy:

```
X_train = np.array(train.iloc[:, :-1].values)

y_train = np.array(train.iloc[:, 1].values)

X_test = np.array(test.iloc[:, :-1].values)

y_test = np.array(test.iloc[:, 1].values)

model = LinearRegression(fit_intercept=True,
normalize=True,copy_X=True,n_jobs=-1)



model.fit(X_train, y_train)



y_pred = model.predict(X_test)
```

```
accuracy = model.score(X_test, y_test)

plt.plot(X_train, model.predict(X_train), color='green')
plt.show()
print(accuracy)
```

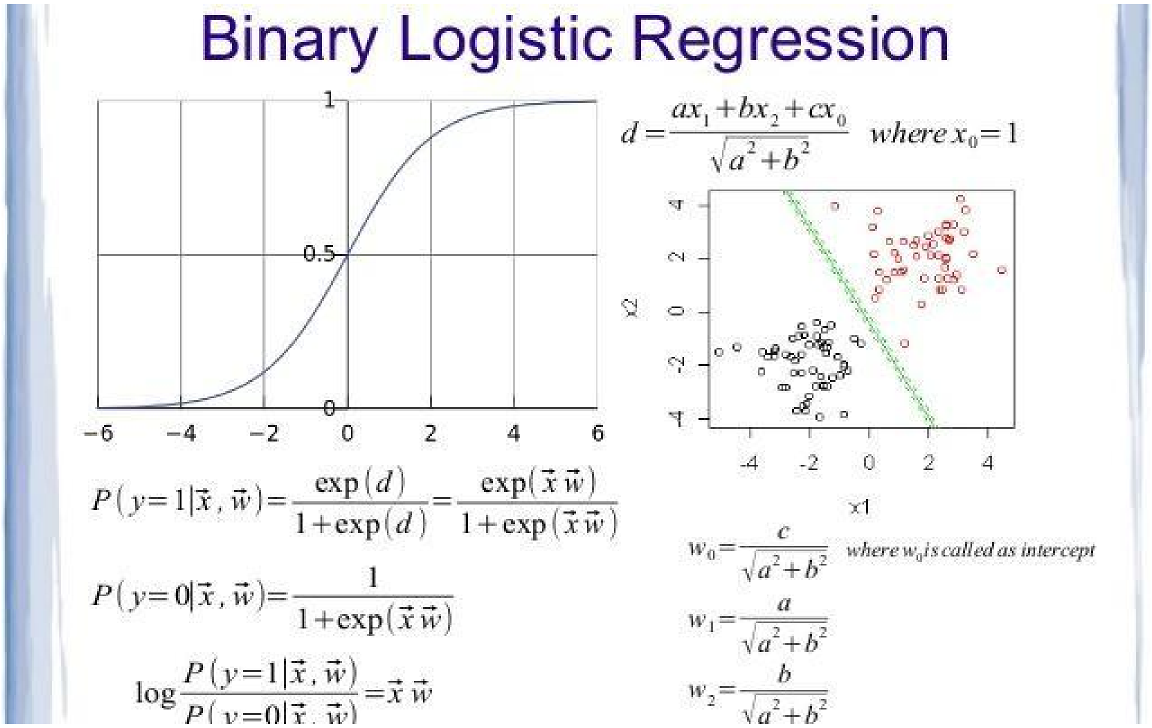


0.9888014444327563

Logistic Regression:

It's a classification algorithm, that is used where the response variable is categorical. The idea of Logistic Regression is to find a relationship between features and probability of particular outcome.

odds= p(x)/(1-p(x)) = probability of event occurrence / probability of not event occurrence



Libraries and data:

import sklearn

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import r2_score

from statistics import mode

train = pd.read_csv("../input/titanic/train.csv")

test = pd.read_csv('../input/titanic/test.csv')

train.head()

OUTPUT:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Model and Accuracy:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,random_state=5)

from sklearn.linear_model import LogisticRegression

#linear_model.LogisticRegression(penalty='l2',dual=False,tol=0.0001,C=1.0,fit_intercept=True,intercept_scaling=1,

#  class_weight=None,random_state=None,solver='warn',max_iter=100,

#   multi_class='warn', verbose=0,warm_start=False, n_jobs=None)



model = LogisticRegression(max_iter = 500000)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = model.score(X_test, y_test)

print(accuracy)
```

OUTPUT:

0.8251121076233184

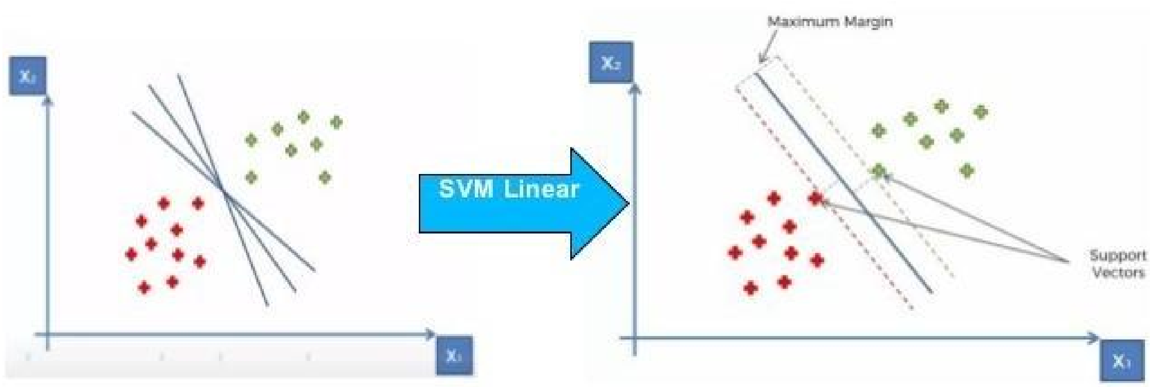
Support Vector Machine:

Data Science

Classification Model : SVM - Linear

Linearly separate the data points

Support Vector Machine" (SVM) is a **supervised machine** learning algorithm which can be used for **both classification or regression challenges**. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordina Then, we perform classification by finding the **hyper-plane that differentiate the two classes** ver well (look at the below snapshot).



Prepared by Manish © IT.mnchoudhary@gmail.com

Support Vector Machines are perhaps one of the most popular and talked about machine learning algorithms.It is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables

Example: One class is linearly separable from the others like if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two dimensional space where each point has two co-ordinates

Libraries and Data:

from sklearn.model_selection import train_test_split

from sklearn.model_selection import cross_val_score

from sklearn.svm import SVC

data_svm = pd.read_csv("../input/svm-classification/UniversalBank.csv")

data_svm.head()

	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
1	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
2	3	39	15	11	94720	1	1.0	1	0	0	0	0	0	0
3	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
4	5	35	8	45	91330	4	1.0	2	0	0	0	0	0	1

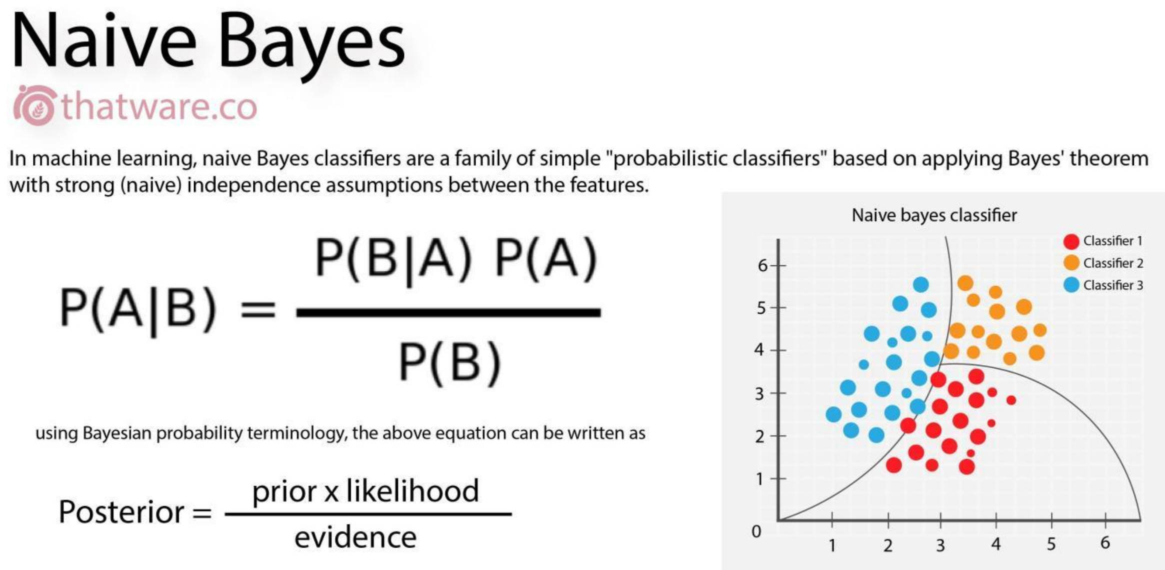
Model and Accuracy:

```
X = data_svm.iloc[:,1:13].values

y = data_svm.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


'''

sklearn.svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='auto_deprecated', coef0=0.0,
shrinking=True,

                    probability=False, tol=0.001, cache_size=200,
class_weight=None,verbose=False,

                    max_iter=-1, decision_function_shape='ovr', random_state=None)

'''


classifier = SVC(kernel = 'rbf', random_state = 0)

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, cv = 10)

accuracies.mean()
```

OUTPUT:

0.7077333333333333

Naive Bayes Algorithm:



A naive Bayes classifier is not a single algorithm, but a family of machine learning algorithms which use probability theory to classify data with an assumption of independence between predictors It is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods

Example: Emails are given and we have to find the spam emails from that.A spam filter looks at email messages for certain key words and puts them in a spam folder if they match.

Libraries and Data:

from sklearn.naive_bayes import GaussianNB

from sklearn.preprocessing import

StandardScalerfrom sklearn.metrics import

accuracy_score

data =
pd.read_csv('../input/classification-suv-dataset/Social_Network_Ads.
csv')

data_nb = data

data_nb.head()

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Model and Accuracy:

X = data_nb.iloc[:,

[2,3]].values y = data_nb.iloc[:,

4].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)

X_test = sc_X.transform(X_test)

'''

#sklearn.naive_bayes.GaussianNB(priors=None, var_smoothing=1e-09)

'''

classifier=GaussianNB()

classifier.fit(X_train,y_train)

y_pred=classifier.predict(X_test)

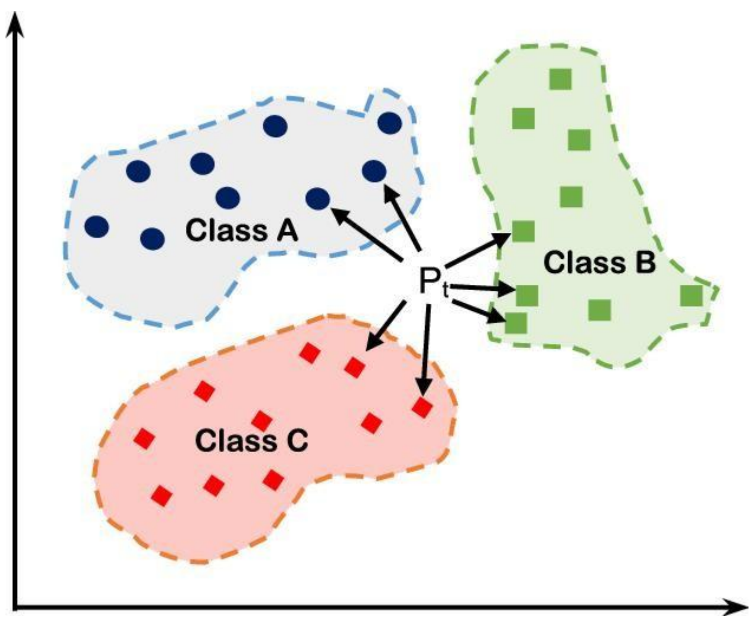
acc=accuracy_score(y_test, y_pred)

```
print(acc)
```

```
0.9125
```

KNN :

KNN does not learn any model. and stores the entire training data set which it uses as its representation. The output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes is taken as the prediction



Libraries and Data :

```
from sklearn.neighbors import KNeighborsClassifier
knn = pd.read_csv("../input/iris/Iris.csv")
knn.head()
```

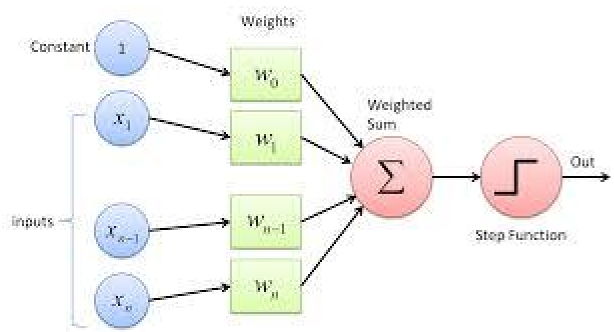
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Model and Accuracy :

```
X = knn.iloc[:, [1,2,3,4]].values
y = knn.iloc[:, 5].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)


'''
sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, weights='uniform', algorithm='auto',
leaf_size=30,
                          p=2, metric='minkowski', metric_params=None,n_jobs=None)
'''

classifier=KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print( acc)
```

1.0

Perceptron:



It is single layer neural network and used for classification

```
from sklearn.linear_model import Perceptron
from sklearn.neighbors import KNeighborsClassifier
p = pd.read_csv("../input/iris/Iris.csv")
p.head()
```

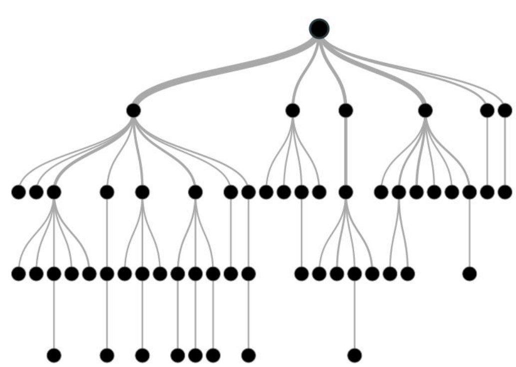
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
X = p.iloc[:, [1,2,3,4]].values
y = p.iloc[:, 5].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
'''
classifier=Perceptron(penalty=None, alpha=0.0001, fit_intercept=True, max_iter=None, tol=None,
shuffle=True,
verbose=0, eta0=1.0, n_jobs=None, random_state=0, early_stopping=False, validation_fraction=0.1,
n_iter_no_change=5, class_weight=None, warm_start=False, n_iter=None)
'''

classifier=Perceptron()
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print(acc)
```

```
0.9666666666666667
```

Decision Tree :

Decision tree algorithm is classification algorithm under supervised machine learning and it is simple to understand and use in data.The idea of Decision tree is to split the big data(root) into smaller(leaves)



```
from sklearn.tree import DecisionTreeClassifier
dt = data
dt.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

```
X = dt.iloc[:, [2,3]].values
y = dt.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)


'''
sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best',
max_depth=None,min_samples_split=2,
                    min_samples_leaf=1,min_weight_fraction_leaf=0.0,max_features=None,
                    random_state=None, max_leaf_nodes=None,min_impurity_decrease=0.0,
                    min_impurity_split=None, class_weight=None,presort=False)
'''


classifier=DecisionTreeClassifier(criterion="entropy",random_state=0)
classifier.fit(X_train,y_train)
y_pred=classifier.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print(acc)
```

0.9

Extra Tree:

```
from sklearn.ensemble import ExtraTreesClassifier
et = data
et.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

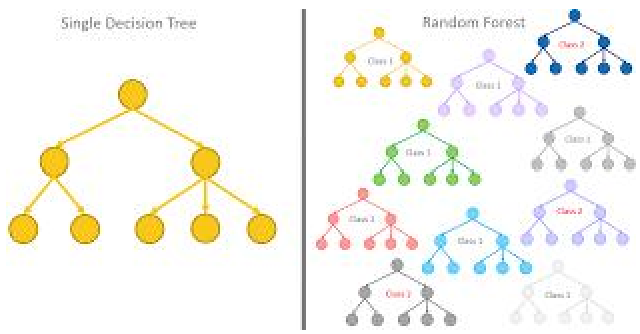
```
X = et.iloc[:, [2,3]].values
y = et.iloc[:, 4].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)


''' ExtraTreesRegressor(n_estimators=10, max_features=32, random_state=0)'''

classifier=ExtraTreesClassifier(criterion="entropy",random_state=0)
classifier.fit(X_train,y_train)
```

```
y_pred=classifier.predict(X_test)
acc=accuracy_score(y_test, y_pred)
print(acc)
```

0.9

Random Forest:

Random forest is collection of tress(forest) and it builds multiple decision trees and merges them together to get a more accurate and stable prediction.It can be used for both classification and regression problems.



Example: Suppose we have a bowl of 100 unique numbers from 0 to 99. We want to select a random sample of numbers from the bowl. If we put the number back in the bowl, it may be selected more than once.

Libraries and Data:

```
from sklearn.ensemble import RandomForestClassifier
rf = pd.read_csv("../input/mushroom-classification/mushrooms.csv")
rf.head()
```

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color	ring-number	ring-type	spore-print-color	populat
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w	o	p	k	s
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w	o	p	n	n
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w	o	p	n	n
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w	o	p	k	s
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w	o	e	n	a

```
X = rf.drop('class', axis=1)
y = rf['class']
X = pd.get_dummies(X)
y = pd.get_dummies(y)
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

```
'''
ensemble.RandomForestClassifier(n_estimators='warn', criterion='gini', max_depth=None,
                    min_samples_split=2, min_samples_leaf=1,min_weight_fraction_leaf=0.0,
                    max_features='auto',max_leaf_nodes=None,min_impurity_decrease=0.0,
                    min_impurity_split=None, bootstrap=True,oob_score=False, n_jobs=None,
```

```
                    random_state=None, verbose=0,warm_start=False, class_weight=None)
'''


model = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=1)
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

1.0