Decision Tree Algorithm.

- The decision tree algorithm is nothing but **supervised machine learning classification and Regression algorithm**.In that we algorthm will classify the feature according to information avalible inbetween feature from the top to bottom in recursive mannaer untill and unless instances,attributs,infomation get over.
- The decision is behave like human mimics and it is easy to interpret.
- The decison having the nature of partitioning the feature accrording to the disimarity between the feature in resursive manner from the top to bottom untill and unless all feature will get the decision or final out in the final label.

Decison Tree structure

- It contains the feature as node and branches works as decision rule and we will select the top most node having the highest information gain and leaf node is an outcome.According the avaliblity of information the partitioning will happen and start to make the sub trees and and it will break down the main tree in the subtreses.That will happen until and unless the information of the became zero.

Attribue selection Measures

- 1.**Gini index** - It is kind of index that will show how the partioning has done accross the all the feature.If there is any wrong with partioning the there will some misclassification that specificaly measure in form of Gini impurity.If the gini index is near by 1 that indicates the higher impurity or higher misclassification happened and when it is close to zero that means the impurity is less and higher chances of getting exact classification.
- 2.**Gini ratio**:- It is nothing the values that are use to avoid the bias in information gain towards an attribue by cartain values.To get the rid of it by adding the values at the denominator to information gain. This is also called split information.It show us that how unformly attribute split the data.It is actually normalize the information gain and help to prevent the missclassification or else biased classification by using some values at the denominator of information gain.
- 3:-**Entropy**:-It is measurement to randomness of the datapoint in the datasets.When we have the high entropy that time highest information gain.
- Information gain is nothing but diffrence between the entropy before and after splitting.
- **Highest Information Gain = entropy at root node - (sum of entropy at the leaf node).**

Types Decision Trees

- 1.Cart
- 2.ID3
    - Here the cart is work for the regression and classification problem that will use the gini index.
    - Here the ID3 is work for the both the classification and not so much for regression by using the information gain.
- To avild the overfitting we having the pruning that will remove the uncessary braches of the tree that will causes the overfitting.
- The overfiting of our model happen beacsue sometime they will gett work with the training data that contains an outliers and decision tree algorithm is sensetive to the outliers.

Let's Start Machine Learning With Decision Tree.

In [206…
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

In [207…
```
data = pd.read_csv('diabetes.csv')
```

In [208…
```
data.head()
```

Out[208…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [209…
```
data.tail()
```

Out[209…

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

In [210…
```
data.shape
```

Out[210…
```
(768, 9)
```

Data Preprocessing.

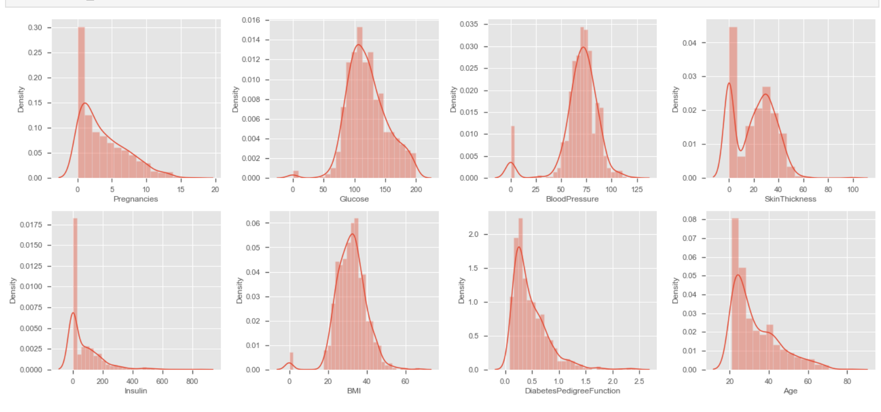
In [211…
```
data.isnull().sum()
```

Out[211…
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [212…
```
len(data.columns)
```

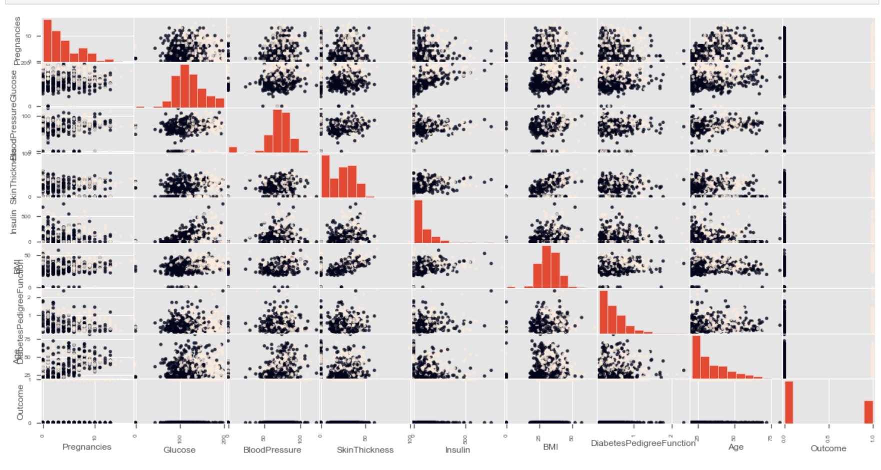
Out[212…
```
9
```

In [213…
```
rows = 2
cols = 4

fig,ax = plt.subplots(nrows=2,ncols=4,figsize=(18,8))

col = data.columns
index=0
for i in range(rows):
    for j in range(cols):
        sns.distplot(data[col[index]],ax=ax[i][j])
        index = index+1
plt.tight_layout()
```



- In this graph if we can see that Glucose,BloodPressure and BMI contains the good information.

In [214…
```
pd.plotting.scatter_matrix(data,c=data['Outcome'], s=20,diagonal='hist',marker='o',figsize=(20,10),alpha=0.8)
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.show()
```



In [215…
```
import seaborn as sns
sns.set(style="ticks", color_codes=True)
sns.pairplot(data,hue='Outcome')
```

Out[215…
```
<seaborn.axisgrid.PairGrid at 0x1faffc3c3d0>
```

Model Building.

In [216…
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
x= data.drop(columns=['Outcome'])
y = data['Outcome']
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=42,stratify=y,train_size=0.70)
```

In [217…
```
x_train.shape,x_test.shape
```

Out[217…
```
((537, 8), (231, 8))
```

In [218…
```
y_train.value_counts()
```

Out[218…
```
0    350
1    187
Name: Outcome, dtype: int64
```

In [219…
```
y_train = y_train.to_numpy()
```

In [220…
```
from imblearn.over_sampling import SMOTE
```

In [221…
```
sm = SMOTE(sampling_strategy=1,random_state=42,k_neighbors=5)
```

In [222…
```
x_train_res,y_train_res = sm.fit_resample(x_train,y_train.ravel())
```

In [223…
```
print(len(y_train_res[y_train_res==0]))
print(len(y_train_res[y_train_res==1]))
```

```
350
350
```

In [224…
```
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV
from sklearn.tree import DecisionTreeRegressor,DecisionTreeClassifier
model = DecisionTreeClassifier()
```

In [225…
```
model_grid = GridSearchCV(model,param_grid={'criterion':['gini','entropy'],'min_samples_leaf':np.arange(1,10,1)
```

In [226…
```
model_grid.fit(x_train_res,y_train_res)
```

Out[226…
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': array([10, 20, 30, 40, 50, 60, 70, 80, 90]),
                         'min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9])},
             scoring='accuracy')
```

In [227…
```
model_cv.best_params_
```

Out[227…
```
{'criterion': 'entropy', 'max_depth': 30, 'min_samples_leaf': 1}
```

In [228…
```
model.set_params
```

Out[228…
```
<bound method BaseEstimator.set_params of DecisionTreeClassifier()>
```

In [229…
```
model_cv.best_score_
```

Out[229…
```
0.7737499999999999
```

Lets try on the best given parameters via Gridsearchcv

In [230…
```
model = DecisionTreeClassifier(criterion='entropy',max_depth=30,min_samples_leaf=1,random_state=42)
```

In [231…
```
model.fit(x_train_res,y_train_res)
```

Out[231…
```
DecisionTreeClassifier(criterion='entropy', max_depth=30, random_state=42)
```

In [232…
```
y_pred = model.predict(x_test)
```

In [233…
```
print('Accuracy :',accuracy_score(y_test,y_pred))
```

```
Accuracy : 0.696969696969697
```

In [234…
```
from sklearn.metrics import mean_squared_error
```

In [235…
```
np.sqrt(mean_squared_error(y_test,y_pred))
```

Out[235…
```
0.5504818825631803
```

In [236…
```
np.std(y_test)
```

Out[236…
```
0.47717332651620464
```

Since the our model is not good than the base model so we will see the Randomserchgrid for that.

RandomSearchcv Approach

In [237…
```
model_Randgrid = RandomizedSearchCV(model,param_distributions={'criterion':['gini','entropy'],'min_samples_leaf
```

In [238…
```
model_Randgrid.fit(x_train_res,y_train_res)
```

Out[238…
```
RandomizedSearchCV(cv=10,
                   estimator=DecisionTreeClassifier(criterion='entropy',
                                                    max_depth=30,
                                                    random_state=42),
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': array([10, 20, 30, 40, 50, 60, 70, 80, 90]),
                                        'min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9])},
                   scoring='accuracy')
```

In [239…
```
model_Randgrid.best_estimator_
```

Out[239…
```
DecisionTreeClassifier(max_depth=30, random_state=42)
```

In [240…
```
model_Randgrid.get_params
```

Out[240…
```
<bound method BaseEstimator.get_params of RandomizedSearchCV(cv=10,
                   estimator=DecisionTreeClassifier(criterion='entropy',
                                                    max_depth=30,
                                                    random_state=42),
                   param_distributions={'criterion': ['gini', 'entropy'],
                                        'max_depth': array([10, 20, 30, 40, 50, 60, 70, 80, 90]),
                                        'min_samples_leaf': array([1, 2, 3, 4, 5, 6, 7, 8, 9])},
                   scoring='accuracy')>
```

Lets try on the best given parameters via RandomSearchcv

In [241…
```
model = DecisionTreeClassifier(criterion='entropy',max_depth=70,random_state=42)
model.fit(x_train_res,y_train_res)
y_pred = model.predict(x_test)
print('Accuracy :',accuracy_score(y_test,y_pred))
from sklearn.metrics import mean_squared_error
print('MSE :- ',np.sqrt(mean_squared_error(y_test,y_pred)))
print('Base Model Error',np.std(y_test))
```

```
Accuracy : 0.696969696969697
MSE :-  0.5504818825631803
Base Model Error 0.47717332651620464
```

Both models are throwing the more MSE than base model error.so this is not appropriate model for this dataset.

To Wheather Model is Overfitted or NOT.

In [242…
```
print('Accuarcy with the training data :-',model.score(x_train,y_train))
print('Accuarcy with the testing data :-',model.score(x_test,y_test))
```

```
Accuarcy with the training data :- 1.0
Accuarcy with the testing data :- 0.696969696969697
```

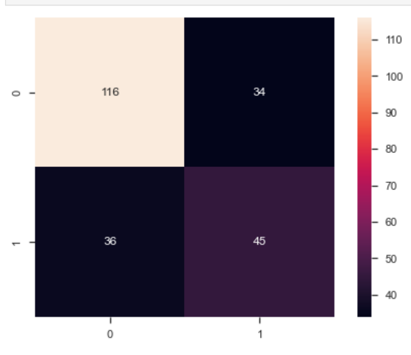
From the above we can easily understand our model well performing on training data and but not performing on the test so from the we can see the model is overfitted.

In [243…
```
from sklearn.metrics import confusion_matrix,mean_squared_error,classification_report
```

In [244…
```
mat = confusion_matrix(y_test,y_pred)
mat
```

Out[244…
```
array([[116,  34],
       [ 36,  45]], dtype=int64)
```

In [245…
```
sns.heatmap(mat,annot=True,fmt='d',square=True)
plt.show()
```



In [246…
```
from sklearn.model_selection import cross_val_score
```

In [247…
```
val_score_mean = cross_val_score(model,x_train,y_train,cv=10).mean()*100
val_score_mean
```

Out[247…
```
70.20964360587001
```

In [248…
```
val_score_max = cross_val_score(model,x_train,y_train,cv=10).max()*100
val_score_max
```

Out[248…
```
79.62962962962963
```

In [249…
```
val_score_min = cross_val_score(model,x_train,y_train,cv=10).min()*100
val_score_min
```

Out[249…
```
62.96296296296296
```

- Since,there is cross_validation moving from 77.41 to 53.22 for k_folds =10.

In [250…
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.77      0.77       150
           1       0.57      0.56      0.56        81

    accuracy                           0.70       231
   macro avg       0.67      0.66      0.67       231
weighted avg       0.70      0.70      0.70       231
```

AUC-ROC CURVE

In [251…
```
from sklearn.metrics import auc,roc_curve
```

In [252…
```
probab=model.predict_proba(x_test)[:,1]
```

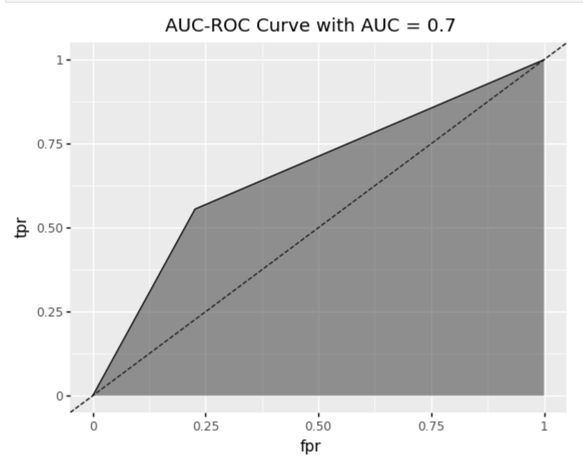
In [253…
```
fpr,tpr,thresh=roc_curve(y_test,probab)
```

In [254…
```
roc_data = pd.DataFrame(dict(fpr=fpr,tpr=tpr))
```

In [255…
```
auc=round(auc(fpr,tpr),1)
```

In [256…
```
from plotnine import aes,geom_abline,geom_area,geom_line,ggplot,ggtitle
```

In [257…
```
ggplot(roc_data,aes(x='fpr',y='tpr'))+geom_abline(linetype='dashed')+geom_area(alpha=0.5)+geom_line()+ggtitle('
```



Out[257…
```
<ggplot: (136094578692)>
```

Thank You.