Decision Tree Method-WORKSHOP PYTHON

DATA SCIENCE INSTITUTE

Introduction to Decision Tree:Recap

- One of the most robust predictive modeling techniques, Decision Tree uses data mining techniques for model building.

- Decision Tree **breaks down a data set into smaller subsets and presents association between target variable(dependent) and independent variables as a tree structure.**

- Final result is a **tree with Decision Nodes and Leaf Nodes.**

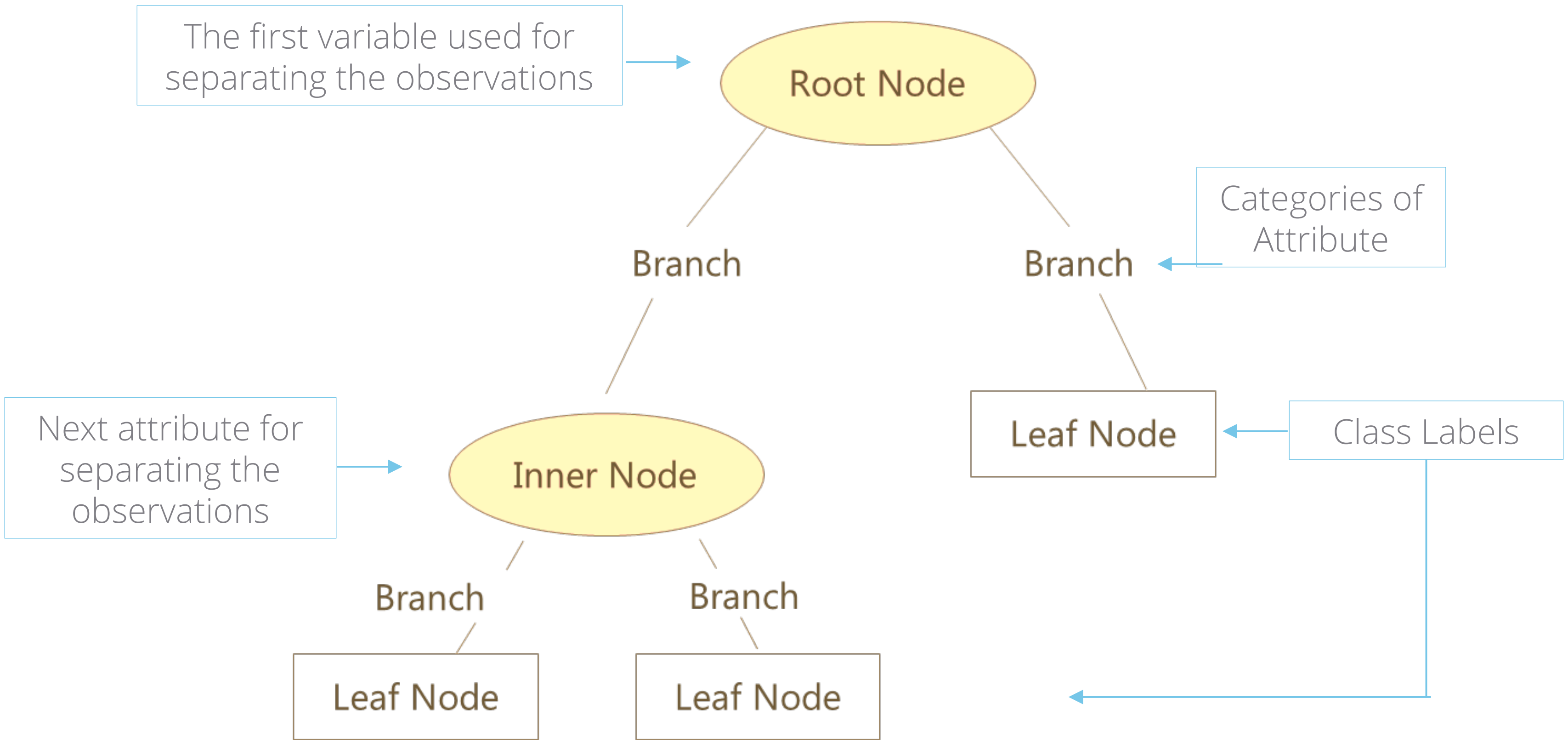
- A decision node has two or more branches and leaf node represents a classification or decision.

Decision Tree – Basic Components

Component	Description	Alternate terms
Root node	Has no incoming edges and zero or more outgoing edges	Parent node
Internal nodes	Each has exactly one incoming edge and two or more outgoing edges	Decision nodes / Child nodes
Leaf node	Each has exactly one incoming edges and no outgoing edges	Terminal nodes
Branches	Categories of attributes	Edges

DATA SCIENCE INSTITUTE

Decision Tree – Basic Components



* Class labels show observations belong to which class. The leaf node also shows Number of observations and Error rate (Actual classification vs classification given by the tree)

DATA SCIENCE INSTITUTE

Entropy

- Entropy measures the homogeneity of a sample. It is used as a parameter for checking the amount of uncertainty associated with a set of probabilities.
- Entropy lies between 0 and 1

  If the sample is completely homogeneous the entropy is 0 and if the sample is equally divided it has entropy of 1
- Entropy can be of two types, for each category and at the variable level
- Entropy of a category is calculated as:

$$-P1 * \log 2(P1) - P2 * \log 2(P2)$$

where,

P1 is the proportion of class 1

P2 is the proportion of class 2

DATA SCIENCE INSTITUTE

Entropy of a Category

Let us consider survey data from three cities depicting shopper's preferred brand

City	Brand A Voters	Brand B Voters	Number of Voters	% of votes for Brand A	% of votes for Brand B
Delhi	90	310	400	22.5%	77.5%
Chennai	10	90	100	10%	90%
Mumbai	100	100	200	50%	50%

Entropy for each city is calculated as:

Delhi:

Chennai:

Mumbai:

Entropy at the Variable Level

- Entropy at the variable level can be derived by **adding weighted averages of all category level entropy values**
- Weights are the **proportion of respondents in each category(here in each city)**

  In the example under consideration,

  Weights for the categories are

Delhi:

Chennai:

Mumbai:

Entropy at the variable level is

Information Gain

- Information Gain is based on the decrease in entropy after a dataset is split on an attribute
- Constructing a decision tree is about finding attribute that returns the highest information gain

- Information gain can be interpreted as ability of reducing the uncertainty (Entropy) and hence increase predictability

Information Gain

City	Brand A Voters	Brand B Voters	Number of Voters	% of votes for Brand A	% of votes for Brand B
Delhi	90	310	400	22.5%	77.5%
Chennai	10	90	100	10%	90%
Mumbai	100	100	200	50%	50%

Entropy for complete sample is calculated as follows:

P1 = (Total Brand A Voters/Total Voters)

P2 = (Total Brand B Voters/Total Voters)

Entropy at the variable level (Weighted average)

Information Gain

DATA SCIENCE INSTITUTE

Information Gain…

- Information Gain value is used to determine which attribute is the "best" – the attribute with most information gain is chosen
- Information gain for a variable is high when that variable has the low entropy at the variable level (Weighted average)
- Low entropy for a variable implies the classification based on that attribute is fairly homogenous , hence this attribute is selected as the first best attribute
- The same process is repeated till all attributes are used as split variables

Case Study – Predicting Loan Defaulters

Background

- The bank possesses demographic and transactional data of its loan customers. If the bank has a robust model to predict defaulters it can undertake better resource allocation.

Objective

- To predict whether the customer applying for the loan will be a defaulter

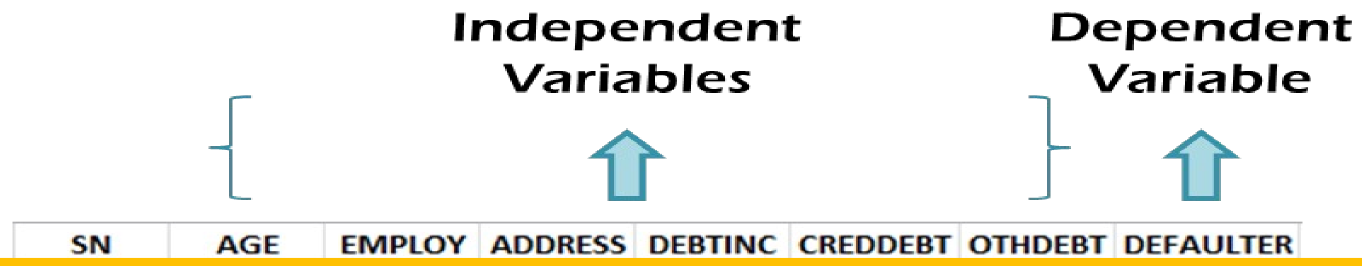
Available Information

- **Sample size is 700**
- **Age group, Years at current address, Years at current employer, Debt to Income Ratio, Credit Card Debts, Other Debts** are the independent variables
- **Defaulter** (=1 if defaulter, 0 otherwise) is the dependent variable

DATA SCIENCE INSTITUTE

Data Snapshot

BANK LOAN



Column	Description	Type	Measurement	Possible Values
SN	Serial Number	Integer	-	-
AGE	Age Groups	Integer	1(<28 years), 2(28-40 years), 3(>40 years)	3
EMPLOY	Number of years customer working at current employer	Integer	-	Positive value
ADDRESS	Number of years customer staying at current address	Integer	-	Positive value
DEBTINC	Debt to Income Ratio	Continuous	-	Positive value
CREDDEBT	Credit to Debit Ratio	Continuous	-	Positive value
OTHDEBT	Other Debt	Continuou	-	Positive value

DATA SCIENCE INSTITUTE

Classification Tree in Python

```
# Importing the Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier,
DecisionTreeRegressor, plot_tree

from sklearn.metrics import confusion_matrix, precision_score,
recall_score, accuracy_score,roc_curve, roc_auc_score
```

- **sklearn.tree** module includes Decision Tree – based models for classification and regression

DATA SCIENCE INSTITUTE

Classification Tree in Python

```
# Importing and Readying the Data for Modeling
bankloan = pd.read_csv("BANK LOAN.csv")


bankloan1 = bankloan.drop(['SN'], axis = 1)

bankloan1['AGE'] = bankloan1['AGE'].astype('ca

bankloan2 = pd.get_dummies(bankloan1)
bankloan2.head()
```

drop() is used to remove unwanted variables.

pd.get_dummies() converts categorical variables into dummy variables. Since AGE is a ... it is

```
# Output
```

	EMPLOY	ADDRESS	DEBTINC	CREDDEBT	OTHDEBT	DEFAULTER	AGE_1	AGE_2	AGE_3
0	17	12	9.3	11.36	5.01	1	0	0	1
1	10	6	17.3	1.36	4.00	0	1	0	0
2	15	14	5.5	0.86	2.17	0	0	1	0
3	15	14	2.9	2.66	0.82	0	0	0	1
4	2	0	17.3	1.79	3.06	1	1	0	0

DATA SCIENCE INSTITUTE

Classification Tree Using Information Gain

```
# Creating Data Partitions
```

```
X = bankloan2.loc[:,bankloan2.columns != 'DEFAULTER']
y = bankloan2.loc[:, 'DEFAULTER']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.30,
                                                    random_state = 999)
```

- **train_test_split()** from sklearn.model_selection is used to split dataset into random train and test sets.
- **test_size** represents the proportion of dataset to be included in the test set.
- **random_state** sets the seed for the random number generator.

Classification Tree Using Information Gain

```
# Classification Tree Using Information Gain
dtcl = DecisionTreeClassifier(criterion='entropy',
                              min_samples_split= int(len(X_train)*.10))
dtcl.fit(X_train, y_train)
```

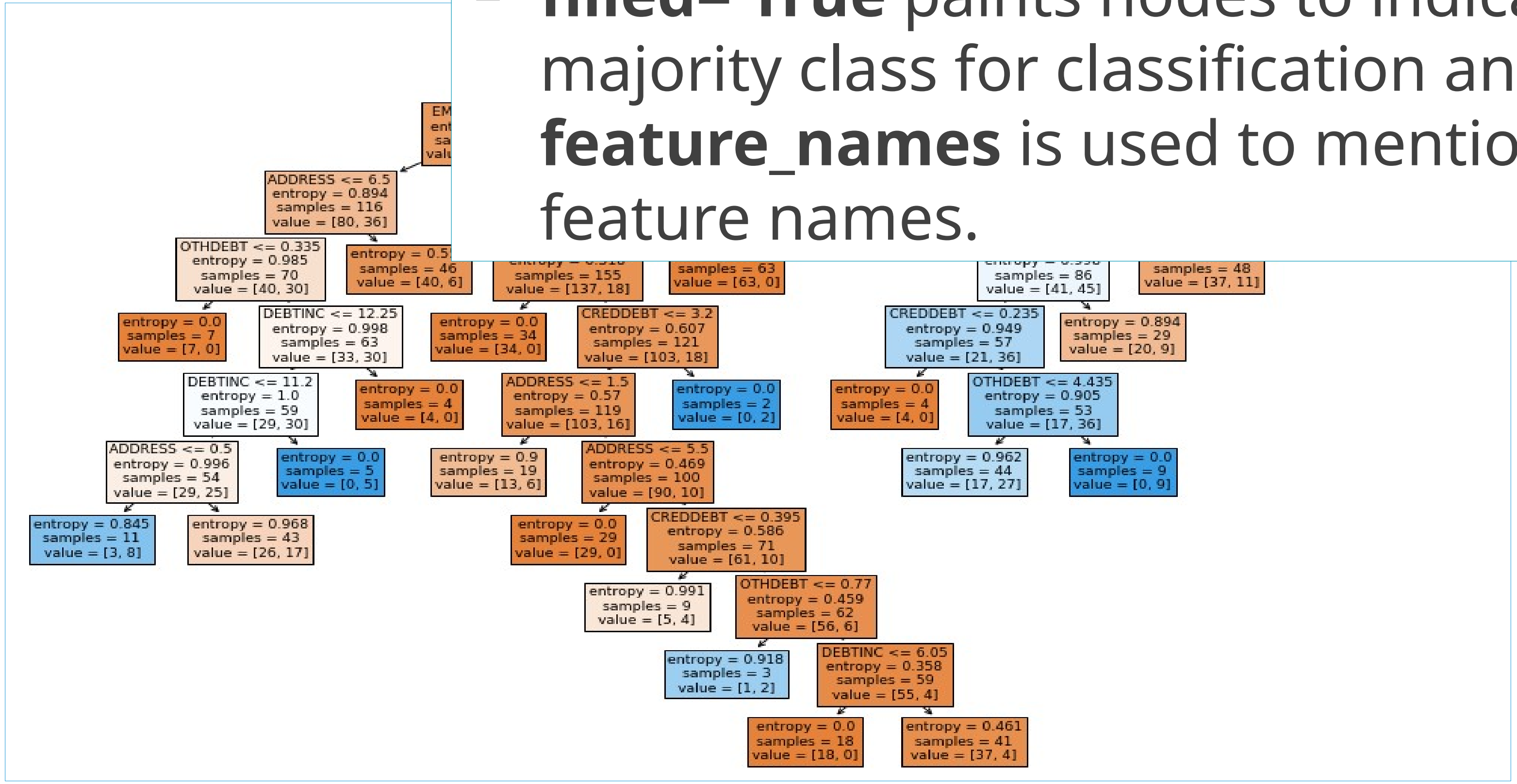
- **DecisionTreeClassifier()** from sklearn.tree fits a classification tree.
- **criterion=** ‘entropy’ specifies the function to measure the split. Default is ‘gini’ for Gini impurity. ‘entropy’ stands for information gain.
- **min_samples_split=** minimum number of samples required to split an internal node. This number is set to be 10% of the sample size.
- The output displays model specifications.

Classification Tree Using Information Gain

Plotting The Tree
```
from sklearn.tree import plot_tree
plt.figure(figsize = (16,10))
plot_tree(dtcl, filled = True, feature_names = list(X.columns))
plt.show();
```

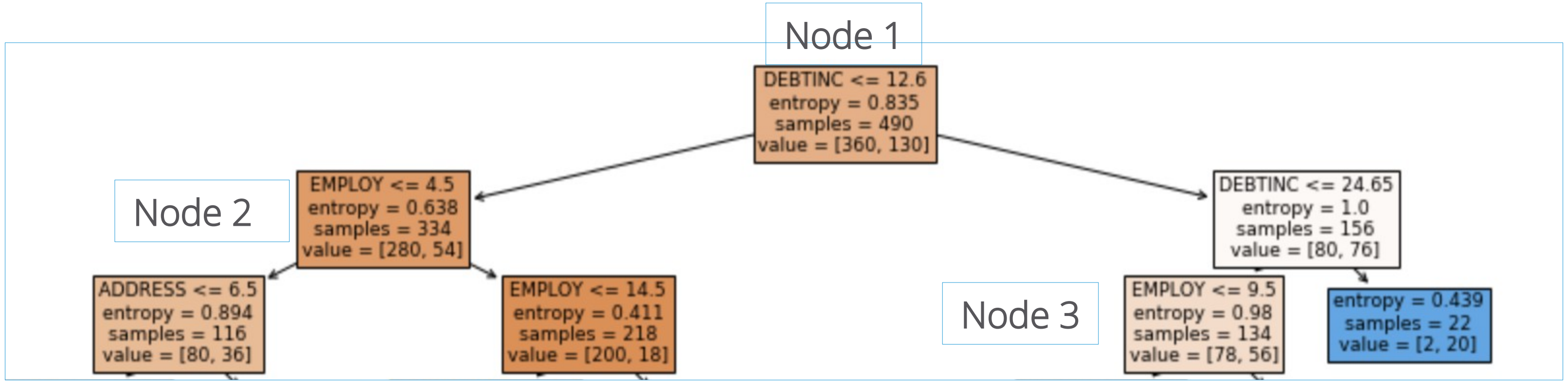
- **plot_tree** is used to plot the decision tree.
- **filled= True** paints nodes to indicate majority class for classification and **feature_names** is used to mention the feature names.

Output



DATA SCIENCE INSTITUTE

Classification Tree Interpretation



Interpretation :

- Due to a large number of continuous predictors, a tree with several nodes and branches is generated.
- Tree starts with all 490 observations (Train set). 360 are non-defaulters (0) and the remaining 130 are defaulters (1).
- DEBTINC is the first split variable, left branch is <=12.6 and right branch is >12.6. 334/490 have DEBTINC<=12.6.
- EMPLOY is the second split on left branch, which further divides 334 obs. into 280 non-defaulters (0) and the remaining 54 as defaulters (1).
- The algorithm progresses till no further variable split is left.

Classification Tree in Python – Prediction

```
# Generating Predictions for the model
```

```
y_pred = dtcl.predict(X_test)
y_pred_probs = dtcl.predict_proba(X_test)

cutoff = 0.3
pred_test = np.where(y_pred_probs[:,1] > cutoff, 1, 0)
pred_test
```

```
# Output
```

```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1])
```

DATA SCIENCE INSTITUTE

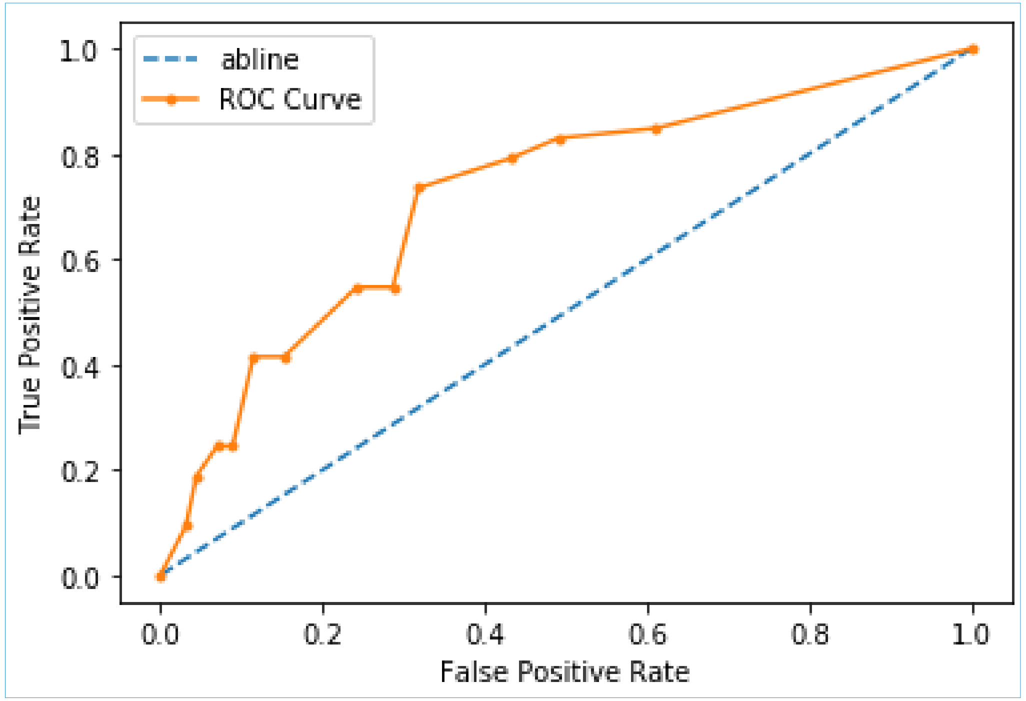
Classification Tree in Python – ROC Curve

```
# Area Under ROC Curve
DTfpr, DTtpr, thresholds = roc_curve(y_test, y_pred_probs[:,1])

abline_probs = [0 for _ in range(len(y_test))]
abline_auc = roc_auc_score(y_test, abline_probs)
abline_fpr, abline_tpr, _ = roc_curve(y_test, abline_probs)

plt.plot(abline_fpr, abline_tpr, linestyle='--', label='abline')
plt.plot(DTfpr, DTtpr, marker='.', label='ROC Curve')
plt.xlabel('False Positive Rate');plt.ylabel('True Positive Rate')
plt.legend(); plt.show()
```

DATA SCIENCE INSTITUTE

Classification Tree in Python – ROC Curve

Output



Plotting The Tree

DATA SCIENCE
INSTITUTE

Classification Tree in Python – Confusion Matrix

```
# Confusion Matrix
confusion_matrix(y_test, pred_test, labels=[0, 1])
array([[107,  50],
       [ 14,  39]], dtype=int64)

accuracy_score(y_test, pred_test)
0.6952380952380952

precision_score(y_test, pred_test)
0.4382022471910125

recall_score(y_test, pred_test)
0.7358490566037735
```

- **accuracy_score() =** number of correct predictions out of total predictions
- **precision_score()** = true positives / (true positives + false positives)
- **recall_score()** also known as 'Sensitivity' = true positives / (true positives + false negatives)

```
# Area Under ROC Curve
auc = roc_auc_score(y_test, y_pred_probs[:,1])
print('AUC: %.3f' % auc)
AUC: 0.720
```

THANK YOU!!

DATA SCIENCE
INSTITUTE