In [1]:
```
# Import Lib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib as mpl
sns.set_style("whitegrid")
```

การนำเข้าข้อมูล Loan

In [2]:
```
df_train = pd.read_csv('Loan_Train.csv')
```

In [3]:
```
df_train
```

Out[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
5	LP001011	Male	Yes	2	Graduate	Yes	5417
6	LP001013	Male	Yes	0	Not Graduate	No	2333
7	LP001014	Male	Yes	3+	Graduate	No	3036
8	LP001018	Male	Yes	2	Graduate	No	4006
9	LP001020	Male	Yes	1	Graduate	No	12841
10	LP001024	Male	Yes	2	Graduate	No	3200
11	LP001027	Male	Yes	2	Graduate	NaN	2500
12	LP001028	Male	Yes	2	Graduate	No	3073
13	LP001029	Male	No	0	Graduate	No	1853
14	LP001030	Male	Yes	2	Graduate	No	1299
15	LP001032	Male	No	0	Graduate	No	4950
16	LP001034	Male	No	1	Not Graduate	No	3596
17	LP001036	Female	No	0	Graduate	No	3510

18	LP001038	Male	Yes	0	Not Graduate	No	4887
19	LP001041	Male	Yes	0	Graduate	NaN	2600
20	LP001043	Male	Yes	0	Not Graduate	No	7660
21	LP001046	Male	Yes	1	Graduate	No	5955
22	LP001047	Male	Yes	0	Not Graduate	No	2600
23	LP001050	NaN	Yes	2	Not Graduate	No	3365
24	LP001052	Male	Yes	1	Graduate	NaN	3717
25	LP001066	Male	Yes	0	Graduate	Yes	9560
26	LP001068	Male	Yes	0	Graduate	No	2799
27	LP001073	Male	Yes	2	Not Graduate	No	4226
28	LP001086	Male	No	0	Not Graduate	No	1442
29	LP001087	Female	No	2	Graduate	NaN	3750
...	...	...	...	...	...	...	...
584	LP002911	Male	Yes	1	Graduate	No	2787
585	LP002912	Male	Yes	1	Graduate	No	4283
586	LP002916	Male	Yes	0	Graduate	No	2297
587	LP002917	Female	No	0	Not Graduate	No	2165
588	LP002925	NaN	No	0	Graduate	No	4750
589	LP002926	Male	Yes	2	Graduate	Yes	2726
590	LP002928	Male	Yes	0	Graduate	No	3000
591	LP002931	Male	Yes	2	Graduate	Yes	6000
592	LP002933	NaN	No	3+	Graduate	Yes	9357
593	LP002936	Male	Yes	0	Graduate	No	3859
594	LP002938	Male	Yes	0	Graduate	Yes	16120
595	LP002940	Male	No	0	Not Graduate	No	3833
596	LP002941	Male	Yes	2	Not Graduate	Yes	6383
597	LP002943	Male	No	NaN	Graduate	No	2987
598	LP002945	Male	Yes	0	Graduate	Yes	9963
599	LP002948	Male	Yes	2	Graduate	No	5780
600	LP002949	Female	No	3+	Graduate	NaN	416

601	LP002950	Male	Yes	0	Not Graduate	NaN	2894
602	LP002953	Male	Yes	3+	Graduate	No	5703
603	LP002958	Male	No	0	Graduate	No	3676
604	LP002959	Female	Yes	1	Graduate	No	12000
605	LP002960	Male	Yes	0	Not Graduate	No	2400
606	LP002961	Male	Yes	1	Graduate	No	3400
607	LP002964	Male	Yes	2	Not Graduate	No	3987
608	LP002974	Male	Yes	0	Graduate	No	3232
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

In [4]:
```
df_train.tail()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

In [5]:
```
df_train.shape
```

Out[5]:
```
(614, 13)
```

Variable Description

- Loan_ID = Unique Loan ID
- Gender = Male/ Female
- Married = Applicant married (Y/N)
- Dependents = Number of dependents
- Education Applicant Education = (Graduate/ Under Graduate)
- Self_Employed = Self employed (Y/N)
- ApplicantIncome = Applicant income
- CoapplicantIncome = Coapplicant income
- LoanAmount = Loan amount in thousands
- Loan_Amount_Term = Term of loan in months
- Credit_History = credit history meets guidelines
- Property_Area = Urban/ Semi Urban/ Rural
- Loan_Status = Loan approved (Y/N)

ประเภทของ data

In [6]:
```
df_train.dtypes
```

Out[6]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

การทำ Feature Engineering

In [7]:
```
df_pre = df_train.copy()
```

In [8]:
```
df_pre.head()
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	

หา Null หรือ NA

In [9]:
```
df_pre.isnull().sum()
```

Out[9]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

Fill NA ด้วยค่า mean

In [10]:
```
df_train.describe()
```

Out[10]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Histo
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.8421
std	6109.041673	2926.248369	85.587325	65.12041	0.3648
min	150.000000	0.000000	9.000000	12.00000	0.0000
25%	2877.500000	0.000000	100.000000	360.00000	1.0000
50%	3812.500000	1188.500000	128.000000	360.00000	1.0000
75%	5795.000000	2297.250000	168.000000	360.00000	1.0000
max	81000.000000	41667.000000	700.000000	480.00000	1.0000

In [11]:
```
df_pre['LoanAmount'].fillna(df_pre['LoanAmount'].mean(),inplace=True)
df_pre['Loan_Amount_Term'].fillna(df_pre['Loan_Amount_Term'].mean(),inplace=True)
df_pre['Credit_History'].fillna(df_pre['Credit_History'].mean(),inplace=True)
df_pre.dropna(inplace=True)
df_pre.isnull().sum()
```

Out[11]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [12]:
```
df_pre.shape
```

Out[12]:
```
(554, 13)
```

Convert String feture to Numerical

Gender

Male = 0 Female = 1

In [13]:
```
df_pre.loc[(df_pre['Gender']=='Male'),'Gender'] = 0
df_pre.loc[(df_pre['Gender']=='Female'),'Gender'] = 1
```

Married

No = 0 Yes = 1

In [14]:
```
df_pre.loc[(df_pre['Married']=='No'),'Married'] = 0.0
df_pre.loc[(df_pre['Married']=='Yes'),'Married'] = 1.0
```

Dependents

Dependents 0 = 0 Dependents 1 = 1 Dependents 2 = 2 Dependents 3+ = 3

In [15]:
```
df_pre.loc[(df_pre['Dependents']=='0'),'Dependents'] = 0.0
df_pre.loc[(df_pre['Dependents']=='1'),'Dependents'] = 1.0
df_pre.loc[(df_pre['Dependents']=='2'),'Dependents'] = 2.0
df_pre.loc[(df_pre['Dependents']=='3+'),'Dependents'] = 3.0
```

Self_Employed

No = 0 Yes = 1

In [16]:
```
df_pre.loc[(df_pre['Self_Employed']=='No'),'Self_Employed'] = 0.0
df_pre.loc[(df_pre['Self_Employed']=='Yes'),'Self_Employed'] = 1.0
```

Education

Not Graduate = 0 Graduate = 1

In [17]:
```
df_pre.loc[(df_pre['Education']=='Not Graduate'),'Education'] = 0.0
df_pre.loc[(df_pre['Education']=='Graduate'),'Education'] = 1.0
```

Property_Area

Urban = 0 Rural = 1 Semiurban = 2

In [18]:
```
df_pre.loc[(df_pre['Property_Area']=='Urban'),'Property_Area'] = 0.
0
df_pre.loc[(df_pre['Property_Area']=='Rural'),'Property_Area'] = 1.
0
df_pre.loc[(df_pre['Property_Area']=='Semiurban'),'Property_Area']
= 2.0
```

Loan_Status

N = 0 Y = 1

In [19]:
```
df_pre.loc[(df_pre['Loan_Status']=='N'),'Loan_Status'] = 0.0
df_pre.loc[(df_pre['Loan_Status']=='Y'),'Loan_Status'] = 1.0
```

In [20]:
```
df_pre.drop('Loan_ID',axis=1,inplace=True)
```

In [21]:
```
df_pre.head(10)
```

Out[21]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicantl
0	0	0	0	1	0	5849	
1	0	1	1	1	0	4583	
2	0	1	0	1	1	3000	
3	0	1	0	0	0	2583	
4	0	0	0	1	0	6000	
5	0	1	2	1	1	5417	
6	0	1	0	0	0	2333	
7	0	1	3	1	0	3036	
8	0	1	2	1	0	4006	
9	0	1	1	1	0	12841	1

Train and Test Split

In [22]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix,
mean_squared_error,r2_score,accuracy_score
import sklearn.metrics as metric
from sklearn.linear_model import LogisticRegression
```

In [23]:
```
X = df_pre.drop('Loan_Status',axis=1)
y = df_pre['Loan_Status']
```

In [24]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=
0.3,random_state=42)
```

In [25]:
```
X_train.head()
```

Out[25]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplica
31	0	0	0	1	0	3167	
516	1	1	2	1	0	2031	
48	1	1	0	1	0	2645	
393	0	1	2	0	0	1993	
210	1	0	0	1	0	10000	

In [26]:
```
y_train.head()
```

Out[26]:
```
31     0
516    1
48     0
393    1
210    0
Name: Loan_Status, dtype: int64
```

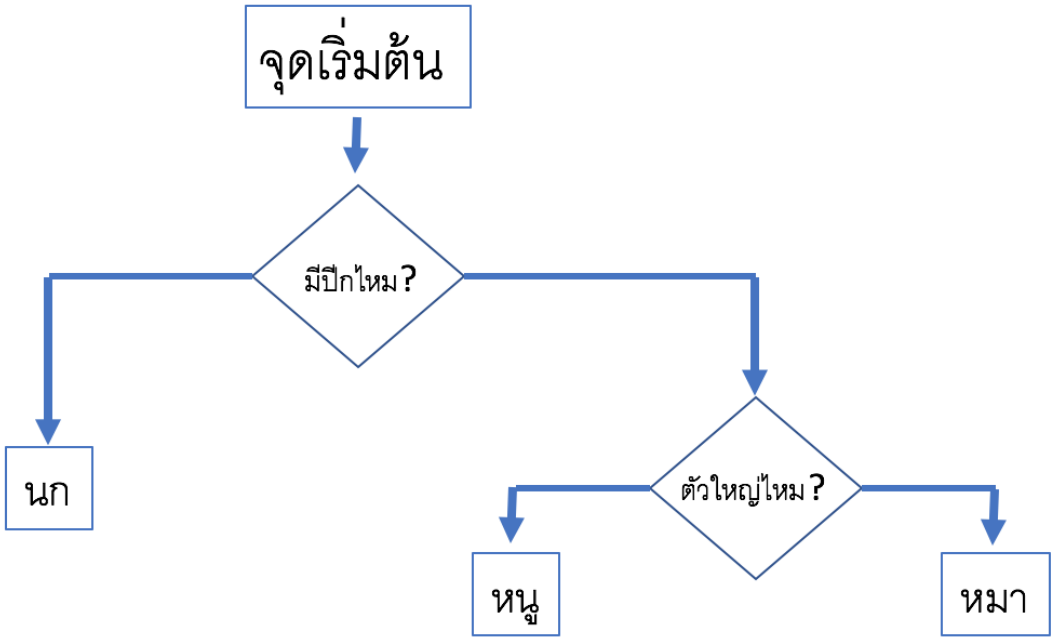
Decision Tree Classifier

อธิบาย

- สมมุติ จะแยกแยะ นก, หมา, หรือ หนู
- เราจะใช้คุณลักษณะในการแยก เช่น จำนวนขา มีปีก ขนาดตัว
- ถ้าเราจะเขียนโปรแกรม เราจะแยกแยะด้วย if else ธรรมดา เพียงแต่ Decision Tree เราไม่จำเป็นต้องมาทำ if else เอง
- สามารถมองเป็นภาพได้ดังนี้

In [27]:
```
from IPython.display import Image
from IPython.core.display import HTML
Image(url= "https://cdn-images-1.medium.com/max/2000/1*am_gg8Av9Ejr5Chs4XKxKw.png")
```

Out[27]:



In [28]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [29]:
```
dtree = DecisionTreeClassifier(criterion='entropy',max_depth = 3, m
in_samples_leaf = 5)
dtree.fit(X_train,y_train)
```

Out[29]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max
_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_st
ate=None,
            splitter='best')
```

In [30]:
```
predictions = dtree.predict(X_test)
predictions_train = dtree.predict(X_train)
```

In [31]:
```
accuracy_train = accuracy_score(y_train, predictions_train)
err_train = mean_squared_error(y_train, predictions_train)
print("Accuracy of Train: %.4f%%" % (accuracy_train * 100.0))
print("Mean Square Error of Train: %.4f%%" % (err_train * 100.0))
accuracy_test = accuracy_score(y_test, predictions)
err_test = mean_squared_error(y_test,predictions)
print("Accuracy of Test: %.4f%%" % (accuracy_test * 100.0))
print("Mean Square Error of Test: %.4f%%" % (err_test * 100.0))
```

```
Accuracy of Train: 79.3282%
Mean Square Error of Train: 20.6718%
Accuracy of Test: 82.6347%
Mean Square Error of Test: 17.3653%
```

In [32]:
```
from sklearn import tree
data = X_train
traget = y_train
dotfile = open('dtree.dot','w')
tree.export_graphviz(dtree,out_file='dtree.dot')
dotfile.close()
```

Xgboost Tree

In [33]:
```
from numpy import loadtxt
from xgboost import XGBClassifier
```

In [34]:
```
xgb = XGBClassifier()
xgb.fit(X_train, y_train)
```

Out[34]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=
1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_s
tep=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators
=100,
       n_jobs=1, nthread=None, objective='binary:logistic', random
_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=True, subsample=1)
```

In [35]:
```
predictions_train = xgb.predict(X_train)
predictions = xgb.predict(X_test)
```

In [36]:
```
accuracy_train = accuracy_score(y_train, predictions_train)
err_train = mean_squared_error(y_train, predictions_train)
print("Accuracy of Train: %.4f%%" % (accuracy_train * 100.0))
print("Mean Square Error of Train: %.4f%%" % (err_train * 100.0))
accuracy_test = accuracy_score(y_test, predictions)
err_test = mean_squared_error(y_test,predictions)
print("Accuracy of Test: %.4f%%" % (accuracy_test * 100.0))
print("Mean Square Error of Test: %.4f%%" % (err_test * 100.0))
```

```
Accuracy of Train: 87.5969%
Mean Square Error of Train: 12.4031%
Accuracy of Test: 84.4311%
Mean Square Error of Test: 15.5689%
```

In [37]:
```
# from sklearn.metrics import classification_report,confusion_matri
x
# print(confusion_matrix(y_test, predictions))
# print(classification_report(y_test, predictions))
```

In [38]:
```
# df_pre.loc[(df_pre['Loan_Status']==1),'Loan_Status'] = 'Y'
# df_pre.loc[(df_pre['Loan_Status']==0),'Loan_Status'] = 'N'
```

In [39]:
```
# df_pre.head()
```

In [40]:
```
# df_pre.to_csv('LoneMATLAB', sep='\t', encoding='utf-8')
```

In [ ]:
```

```