In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('E:/New folder/train.csv')
```

Out[2]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0

Set index

In [3]:
```
data.set_index(data['Loan_ID'], inplace=True)
```

In [4]:

Out[4]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loa
Loan_ID								
LP001002	Male	No	0	Graduate	No	5849	0.0	
LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
LP001008	Male	No	0	Graduate	No	6000	0.0	
LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	
LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	
LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	
LP001018	Male	Yes	2	Graduate	No	4006	1526.0	
LP001020	Male	Yes	1	Graduate	No	12841	10968.0	
LP001024	Male	Yes	2	Graduate	No	3200	700.0	
LP001027	Male	Yes	2	Graduate	NaN	2500	1840.0	
LP001028	Male	Yes	2	Graduate	No	3073	8106.0	
LP001029	Male	No	0	Graduate	No	1853	2840.0	
LP001030	Male	Yes	2	Graduate	No	1299	1086.0	
LP001032	Male	No	0	Graduate	No	4950	0.0	
LP001034	Male	No	1	Not Graduate	No	3596	0.0	
LP001036	Female	No	0	Graduate	No	3510	0.0	
LP001038	Male	Yes	0	Not Graduate	No	4887	0.0	
LP001041	Male	Yes	0	Graduate	NaN	2600	3500.0	

Remove NaN

In [5]:

```
data['Credit_History'].fillna((data['Credit_History'].median()), inplace=True)
data['LoanAmount'].fillna((data['LoanAmount'].median()), inplace=True)
data['Loan_Amount_Term'].fillna((data['Loan_Amount_Term'].median()), inplace=True)
data['Gender'].fillna((data['Gender'].mode()[0]), inplace=True)
data['Married'].fillna((data['Married'].mode()[0]), inplace=True)
data['Dependents'].fillna((data['Dependents'].mode()[0]), inplace=True)
```

In [6]:
```
data.head(30)
```

Out[6]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Lo
Loan_ID								
LP001002	Male	No	0	Graduate	No	5849	0.0	
LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
LP001008	Male	No	0	Graduate	No	6000	0.0	
LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	
LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	
LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	
LP001018	Male	Yes	2	Graduate	No	4006	1526.0	
LP001020	Male	Yes	1	Graduate	No	12841	10968.0	
LP001024	Male	Yes	2	Graduate	No	3200	700.0	
LP001027	Male	Yes	2	Graduate	No	2500	1840.0	
LP001028	Male	Yes	2	Graduate	No	3073	8106.0	
LP001029	Male	No	0	Graduate	No	1853	2840.0	
LP001030	Male	Yes	2	Graduate	No	1299	1086.0	
LP001032	Male	No	0	Graduate	No	4950	0.0	
LP001034	Male	No	1	Not Graduate	No	3596	0.0	
LP001036	Female	No	0	Graduate	No	3510	0.0	
LP001038	Male	Yes	0	Not Graduate	No	4887	0.0	
LP001041	Male	Yes	0	Graduate	No	2600	3500.0	
LP001043	Male	Yes	0	Not Graduate	No	7660	0.0	
LP001046	Male	Yes	1	Graduate	No	5955	5625.0	
LP001047	Male	Yes	0	Not Graduate	No	2600	1911.0	
LP001050	Male	Yes	2	Not Graduate	No	3365	1917.0	
LP001052	Male	Yes	1	Graduate	No	3717	2925.0	
LP001066	Male	Yes	0	Graduate	Yes	9560	0.0	
LP001068	Male	Yes	0	Graduate	No	2799	2253.0	
LP001073	Male	Yes	2	Not Graduate	No	4226	1040.0	
LP001086	Male	No	0	Not Graduate	No	1442	0.0	
LP001087	Female	No	2	Graduate	No	3750	2083.0	

In [7]:
```
[illegible]
```

Out[7]:
```
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 [8]:
```
[illegible]
```

Out[8]:
```
360.0    526
180.0     44
480.0     15
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

Convert categorical data to numerical

In [9]:
```
[illegible]
```

In [10]:
```
[illegible]
```

In [11]:
```
[illegible]
```

In [12]:
```
[illegible]
```

In [ ]:

In [13]:
```
for col in cat_data:
    [illegible]
```

In [14]:

Out[14]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	Loa
Loan_ID								
LP001002	1	0	0	0	0	5849	0.0	
LP001003	1	1	1	0	0	4583	1508.0	
LP001005	1	1	0	0	1	3000	0.0	
LP001006	1	1	0	1	0	2583	2358.0	
LP001008	1	0	0	0	0	6000	0.0	
LP001011	1	1	2	0	1	5417	4196.0	
LP001013	1	1	0	1	0	2333	1516.0	
LP001014	1	1	3	0	0	3036	2504.0	
LP001018	1	1	2	0	0	4006	1526.0	
LP001020	1	1	1	0	0	12841	10968.0	

Split the data into training and testing data

In [15]:

```
loan_x = data.iloc[:,1:-1]
```

In [16]:

Out[16]:

	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmoun
Loan_ID							
LP001002	0	0	0	0	5849	0.0	128.(
LP001003	1	1	0	0	4583	1508.0	128.(
LP001005	1	0	0	1	3000	0.0	66.(
LP001006	1	0	1	0	2583	2358.0	120.(
LP001008	0	0	0	0	6000	0.0	141.(
...	...	...	...	...	...	...	..
LP002978	0	0	0	0	2900	0.0	71.(
LP002979	1	3	0	0	4106	0.0	40.(
LP002983	1	1	0	0	8072	240.0	253.(
LP002984	1	2	0	0	7583	0.0	187.(
LP002990	0	0	0	1	4583	0.0	133.(

614 rows × 10 columns

```
In [17]:
```

```
Out[17]: Loan_ID
         LP001002    1
         LP001003    0
         LP001005    1
         LP001006    1
         LP001008    1
                    ..
         LP002978    1
         LP002979    1
         LP002983    1
         LP002984    1
         LP002990    0
         Name: Loan_Status, Length: 614, dtype: int32
```

```
In [18]:
```

```
In [19]:
```

```
In [20]: print(X_train.shape)
```

```
(491, 10)
(123, 10)
```

Ensemble with KNN

```
In [21]:
```

```
In [22]:
```

```
In [23]:
```

```
In [24]:
```

```
Out[24]: BaggingClassifier(base_estimator=KNeighborsClassifier(algorithm='auto',
                                                               leaf_size=30,
                                                               metric='minkowski',
                                                               metric_params=None,
                                                               n_jobs=None,
                                                               n_neighbors=5, p=2,
                                                               weights='uniform'),
                           bootstrap=True, bootstrap_features=False, max_features=1.0,
                           max_samples=1.0, n_estimators=10, n_jobs=None,
                           oob_score=False, random_state=None, verbose=0,
                           warm_start=False)
```

```
In [25]:
```

```
In [26]:
```

```
Out[26]: 0.6178861788617886
```

```
In [27]:
```

```
In [28]:
```

```
Out[28]: array([[ 4, 36],
                [11, 72]], dtype=int64)
```

There are 42 false predicted values, with 65.84% accuracy

Only KNN

```
In [29]:

In [30]:

In [31]:

Out[31]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')

In [32]:

In [33]:

Out[33]: 0.6016260162601627

In [34]: from sklearn.metrics import confusion_matrix

Out[34]: array([[ 4, 36],
                [13, 70]], dtype=int64)
```

Same accuracy and confusion matrix found

Only Bagging

```
In [35]: from sklearn.ensemble import BaggingClassifier
         bag = BaggingClassifier()
         bag.fit(X_train, y_train)
         print('Accuracy Score: ', bag.score(X_test, y_test))
         y_pred2 = bag.predict(X_test)
         from sklearn.metrics import confusion_matrix

         Accuracy Score:  0.7967479674796748
         Confusion Matrix:  [[25 15]
          [10 73]]
```

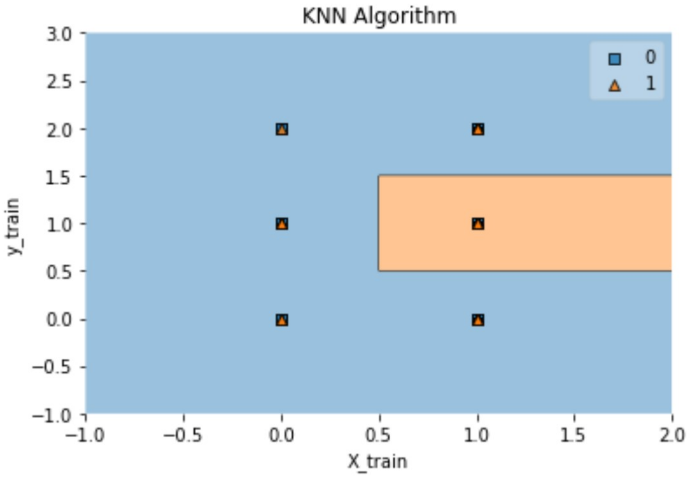
Bagging Classifier gave 19 wrong predictions, with 84.5% accuracy

KNN Algorithm Visualization

```
In [36]:

Out[36]: Index(['Married', 'Dependents', 'Education', 'Self_Employed',
                'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
                'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
               dtype='object')

In [37]: from mlxtend.plotting import plot_decision_regions
         X_train=X_train[['Credit_History','Property_Area']]
         print(X_train.shape)

         (491, 2)
         (491,)
```

In [38]:
```
knn=KNeighborsClassifier()
knn.fit(X_train, y_train)
plot_decision_regions(X_train.to_numpy(), y_train.to_numpy(), clf=knn)
plt.xlabel('X_train')
plt.ylabel('y_train')
plt.title('KNN Algorithm')
```

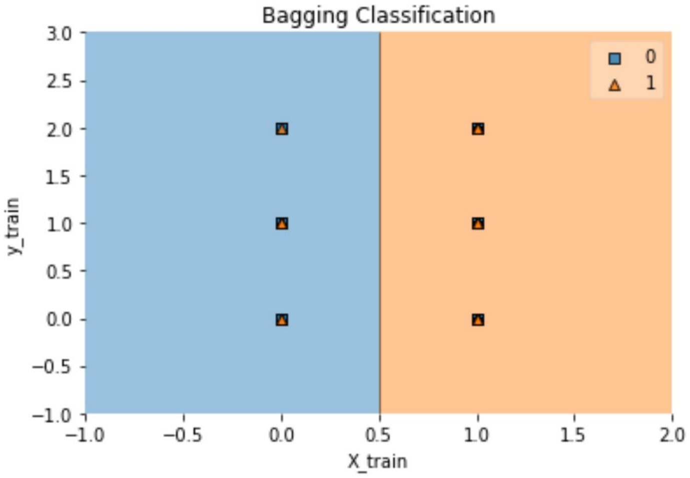
```
C:\Users\SR1407SM1106\AppData\Local\Continuum\anaconda3\lib\site-packages\mlxten
d\plotting\decision_regions.py:249: MatplotlibDeprecationWarning: Passing unsupp
orted keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



Bagging Classifier Visualization

In [39]:
```
bag = BaggingClassifier()
bag.fit(X_train, y_train)
plot_decision_regions(X_train.to_numpy(), y_train.to_numpy(), clf=bag)
plt.xlabel('X_train')
plt.ylabel('y_train')
plt.title('Bagging Classification')
```

```
C:\Users\SR1407SM1106\AppData\Local\Continuum\anaconda3\lib\site-packages\mlxten
d\plotting\decision_regions.py:249: MatplotlibDeprecationWarning: Passing unsupp
orted keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



Decision Tree Classifier

In [40]:
```
X_train, X_test, y_train, y_test = train_test_split(loan_x, loan_y, train_size=0.8
```

In [41]:

In [42]:

Out[42]:

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

In [43]:

Out[43]:

```
0.6829268292682927
```

In [44]:

```
preed_y = dt.predict(X_test)
from sklearn.metrics import confusion_matrix
```

Out[44]:

```
array([[21, 18],
       [21, 63]], dtype=int64)
```

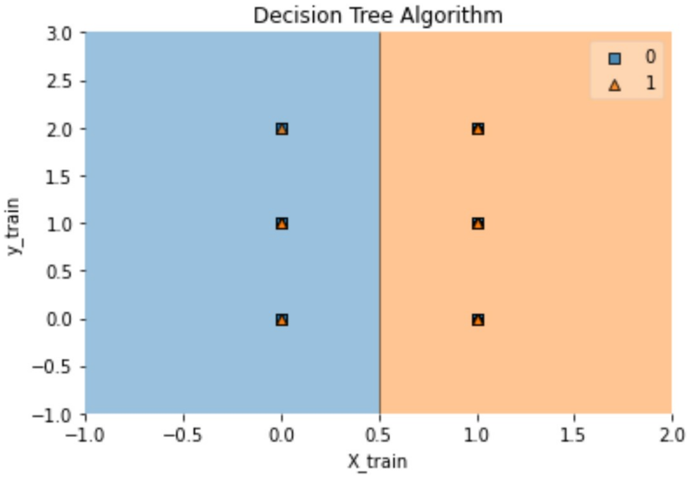
Score: 69.9%

Errors: 37 wrong predictions

In [45]:

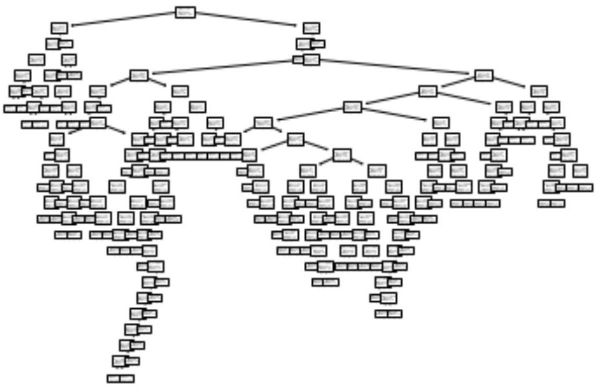
```
from mlxtend.plotting import plot_decision_regions
X_train_plot = X_train[['Credit_History','Property_Area']]
dt = DecisionTreeClassifier()
dt.fit(X_train_plot, y_train)
plot_decision_regions(X_train_plot.to_numpy(), y_train.to_numpy(), clf=dt)
plt.xlabel('X_train')
plt.ylabel('y_train')
plt.title('Decision Tree Algorithm')
```

```
C:\Users\SR1407SM1106\AppData\Local\Continuum\anaconda3\lib\site-packages\mlxten
d\plotting\decision_regions.py:249: MatplotlibDeprecationWarning: Passing unsupp
orted keyword arguments to axis() will raise a TypeError in 3.3.
  ax.axis(xmin=xx.min(), xmax=xx.max(), y_min=yy.min(), y_max=yy.max())
```



In [47]:

```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
tree.plot_tree(clf)
```



In [ ]:

In [ ]:

In [ ]: