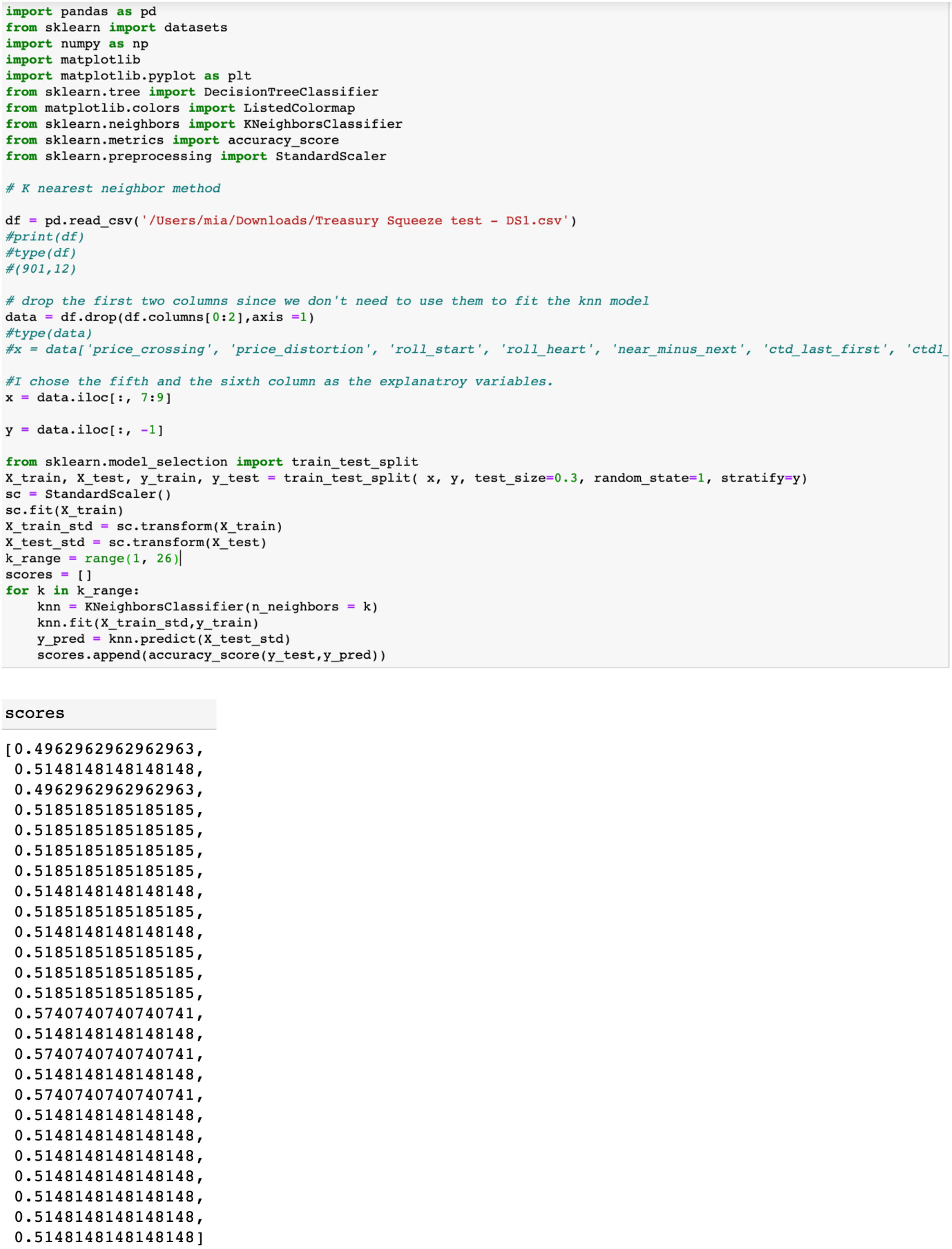
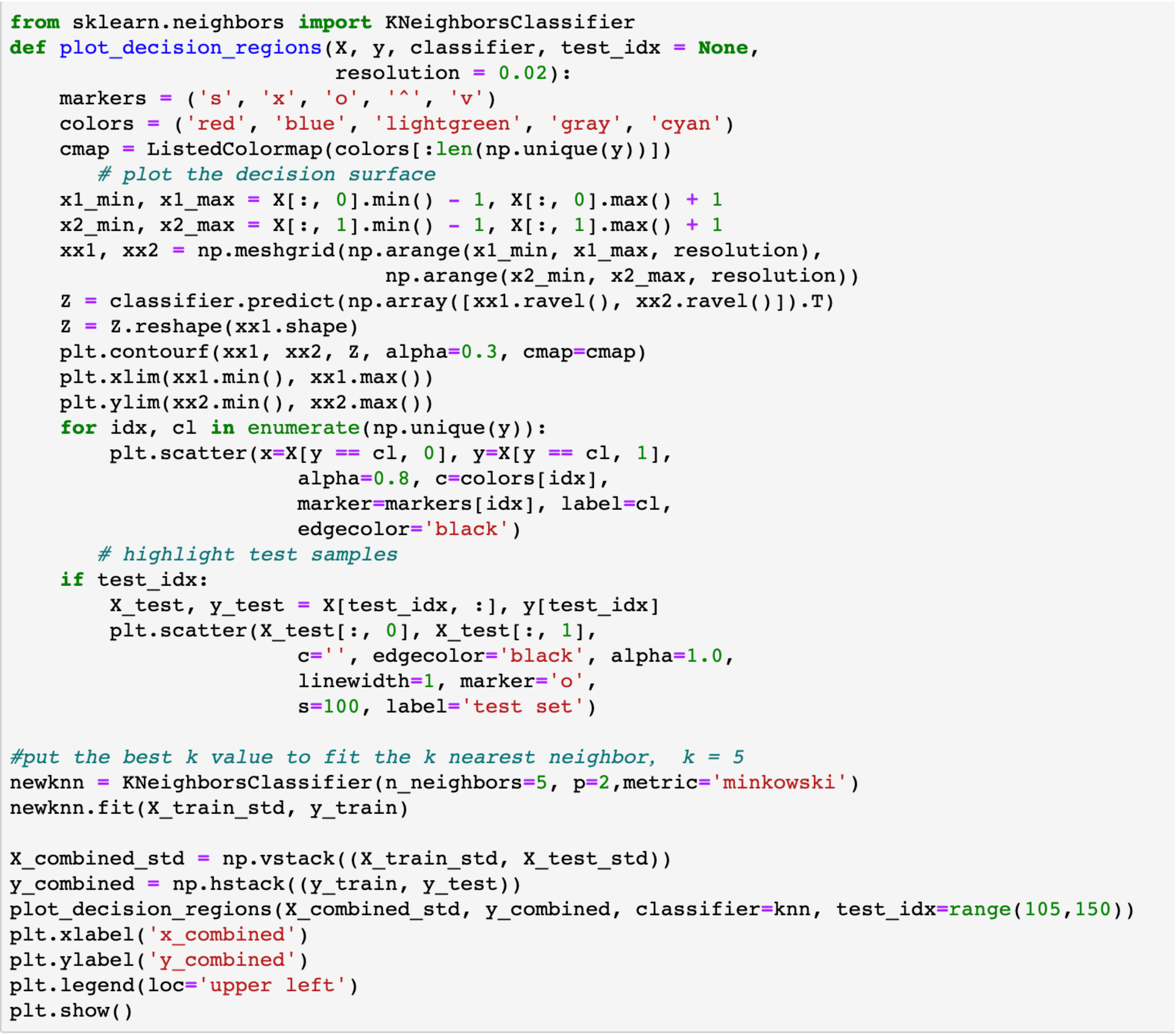


```python
import pandas as pd
from sklearn import datasets
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from matplotlib.colors import ListedColormap
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

# K nearest neighbor method

df = pd.read_csv('/Users/mia/Downloads/Treasury Squeeze test - DS1.csv')
#print(df)
#type(df)
#(901,12)

# drop the first two columns since we don't need to use them to fit the knn model
data = df.drop(df.columns[0:2],axis =1)
#type(data)
#x = data['price_crossing', 'price_distortion', 'roll_start', 'roll_heart', 'near_minus_next', 'ctd_last_first', 'ctdl_

#I chose the fifth and the sixth column as the explanatroy variables.
x = data.iloc[:, 7:9]

y = data.iloc[:, -1]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.3, random_state=1, stratify=y)
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
k_range = range(1, 26)
scores = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    knn.fit(X_train_std,y_train)
    y_pred = knn.predict(X_test_std)
    scores.append(accuracy_score(y_test,y_pred))
```

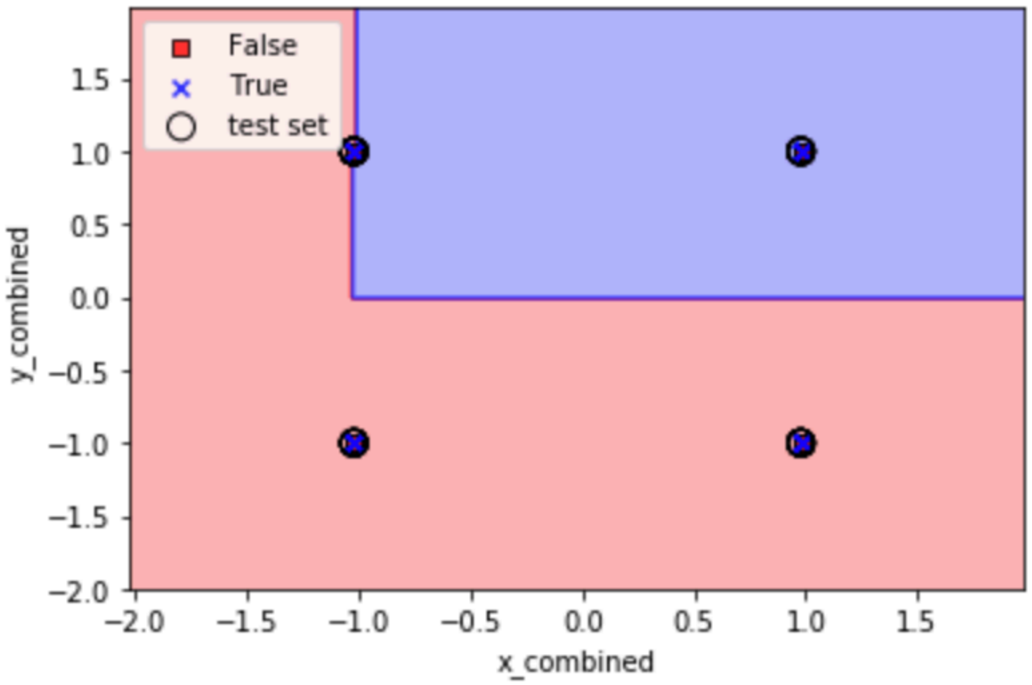
```python
scores
```

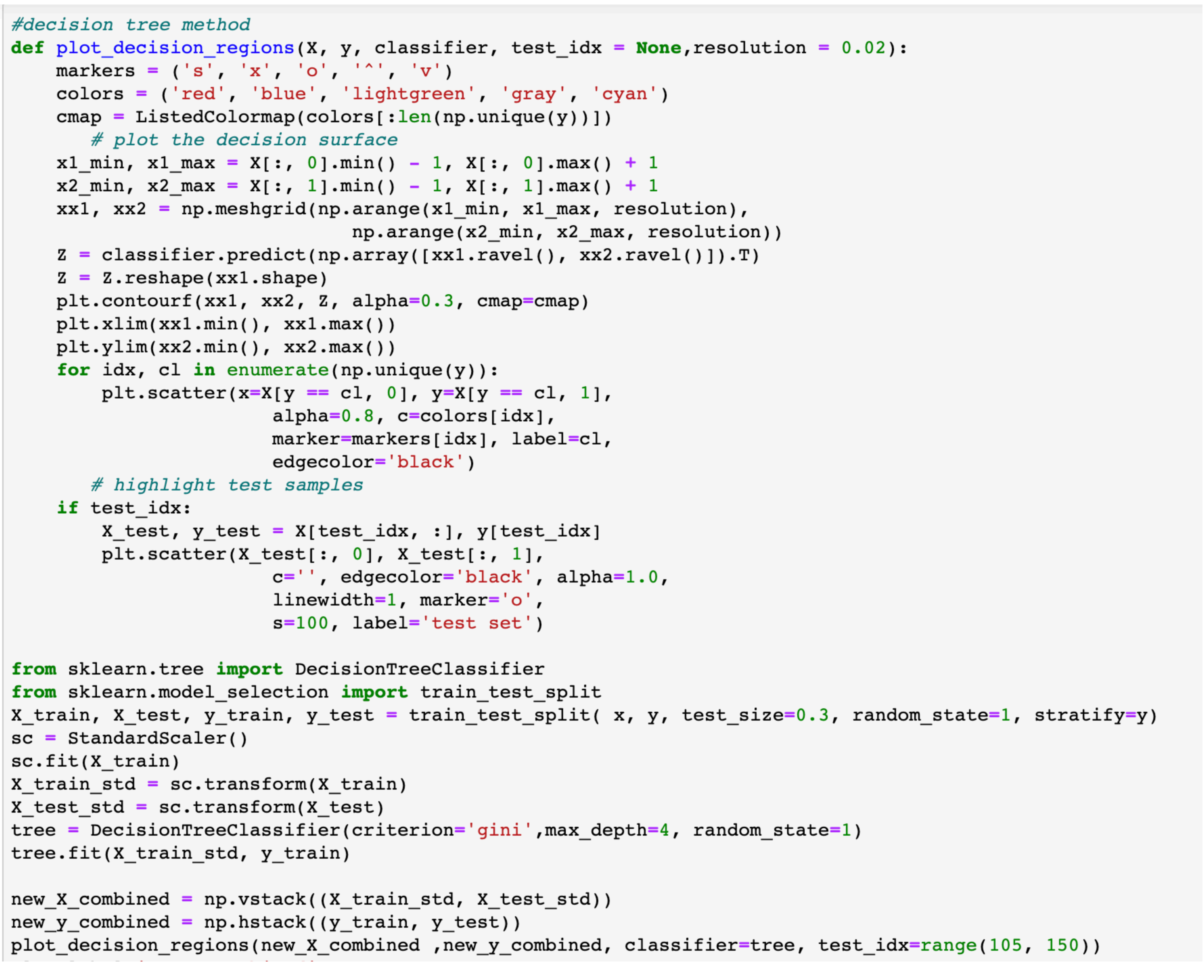
```
[0.4962962962962963,
 0.5148148148148148,
 0.4962962962962963,
 0.5185185185185185,
 0.5185185185185185,
 0.5185185185185185,
 0.5185185185185185,
 0.5148148148148148,
 0.5185185185185185,
 0.5148148148148148,
 0.5185185185185185,
 0.5185185185185185,
 0.5185185185185185,
 0.5740740740740741,
 0.5148148148148148,
 0.5740740740740741,
 0.5148148148148148,
 0.5740740740740741,
 0.5148148148148148,
 0.5148148148148148,
 0.5148148148148148,
 0.5148148148148148,
 0.5148148148148148,
 0.5148148148148148,
 0.5148148148148148]
```



```python
from sklearn.neighbors import KNeighborsClassifier
def plot_decision_regions(X, y, classifier, test_idx = None,
                          resolution = 0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
       # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                              np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                       alpha=0.8, c=colors[idx],
                       marker=markers[idx], label=cl,
                       edgecolor='black')
       # highlight test samples
    if test_idx:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1],
                       c='', edgecolor='black', alpha=1.0,
                       linewidth=1, marker='o',
                       s=100, label='test set')

#put the best k value to fit the k nearest neighbor,  k = 5
newknn = KNeighborsClassifier(n_neighbors=5, p=2,metric='minkowski')
newknn.fit(X_train_std, y_train)

X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined_std, y_combined, classifier=knn, test_idx=range(105,150))
plt.xlabel('x_combined')
plt.ylabel('y_combined')
plt.legend(loc='upper left')
plt.show()
```

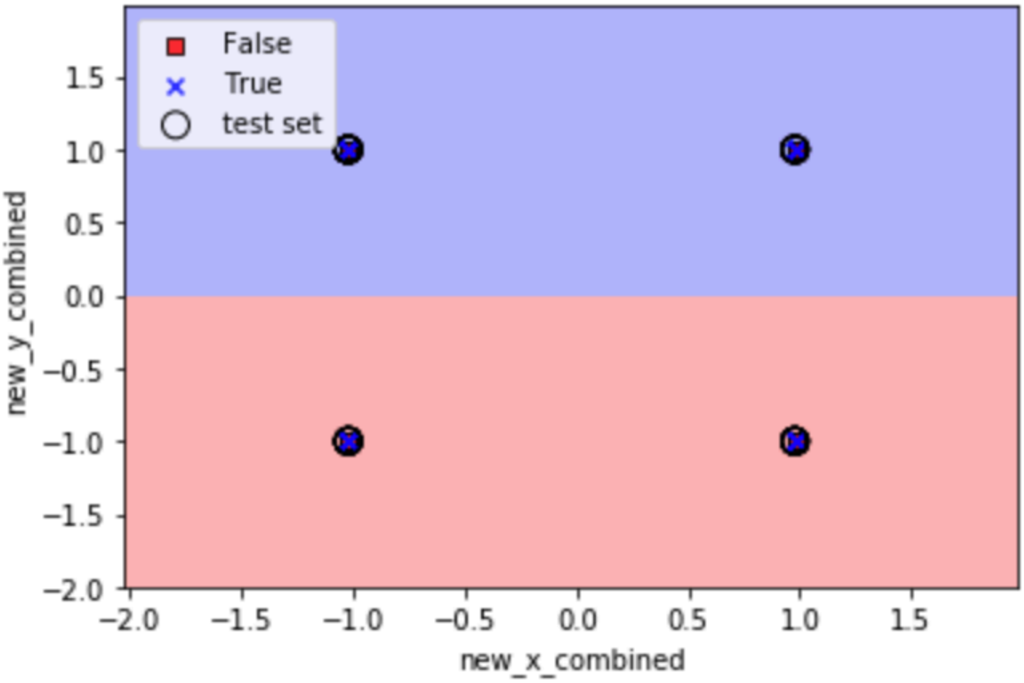
The above part we plug in the best k value and get the plot. The best k value creates the highest accuracy. The plot result is shown below:

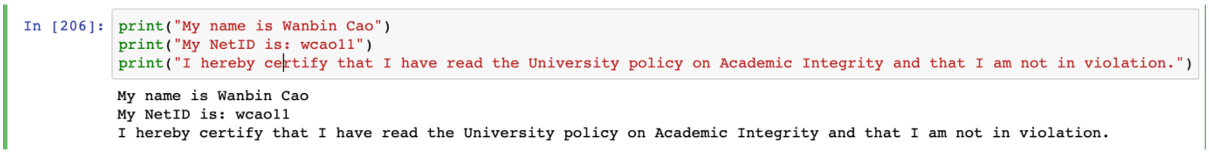




```
#decision tree method
def plot_decision_regions(X, y, classifier, test_idx = None,resolution = 0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
       # plot the decision surface
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.3, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],
                    alpha=0.8, c=colors[idx],
                    marker=markers[idx], label=cl,
                    edgecolor='black')
       # highlight test samples
    if test_idx:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1],
                    c='', edgecolor='black', alpha=1.0,
                    linewidth=1, marker='o',
                    s=100, label='test set')

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( x, y, test_size=0.3, random_state=1, stratify=y)
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
tree = DecisionTreeClassifier(criterion='gini',max_depth=4, random_state=1)
tree.fit(X_train_std, y_train)

new_X_combined = np.vstack((X_train_std, X_test_std))
new_y_combined = np.hstack((y_train, y_test))
plot_decision_regions(new_X_combined ,new_y_combined, classifier=tree, test_idx=range(105, 150))
```

We use the decision tree method above and get the plot below.





```
In [206]: print("My name is Wanbin Cao")
          print("My NetID is: wcao11")
          print("I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.")

          My name is Wanbin Cao
          My NetID is: wcao11
          I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.
```

My github link is