Section 1: Monitoring changes in land cover using satellite images

In [1]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import matplotlib

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix

%matplotlib inline
```

In [13]:

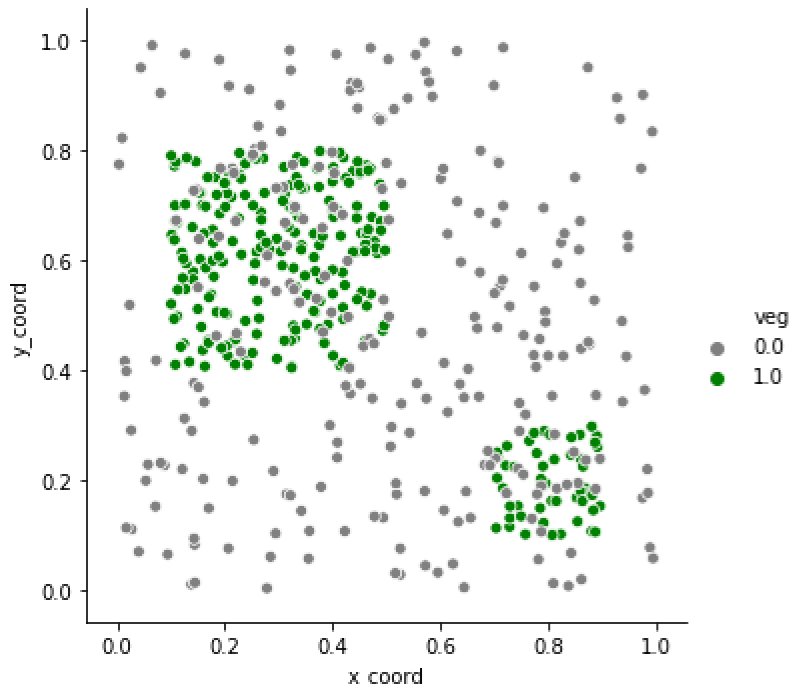
```
df = pd.read_csv('dataset_2.txt', header = None,\
                 names = ['x_coord', 'y_coord', 'veg'])
df.head()
```

Out[13]:

	x_coord	y_coord	veg
0	0.266809	0.688130	1.0
1	0.100046	0.520933	1.0
2	0.158702	0.436935	1.0
3	0.174504	0.538224	1.0
4	0.258707	0.615527	1.0

In [14]:
```
sns.relplot(x='x_coord',y='y_coord',hue='veg', palette=['gray','green'],data=df)
```

Out[14]: <seaborn.axisgrid.FacetGrid at 0x26450638e08>



In [15]:
```
x = df[['x_coord','y_coord']]
y = df[['veg']]
```

In [16]:
```
x.head()
```

Out[16]:

	x_coord	y_coord
0	0.266809	0.688130
1	0.100046	0.520933
2	0.158702	0.436935
3	0.174504	0.538224
4	0.258707	0.615527

In [17]:
```
y.head()
```

Out[17]:

	veg
0	1.0
1	1.0
2	1.0
3	1.0
4	1.0

In [18]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
```

In [19]:

```
x_train
```

Out[19]:

	x_coord	y_coord
177	0.352921	0.538119
255	0.574219	0.348988
418	0.183953	0.462839
4	0.258707	0.615527
142	0.200947	0.741959
...	...	...
220	0.823703	0.102073
129	0.214231	0.634733
139	0.237338	0.719056
103	0.341724	0.619819
467	0.503510	0.965933

400 rows × 2 columns

In [20]:

```
y_train
```

Out[20]:

	veg
177	1.0
255	0.0
418	0.0
4	1.0
142	1.0
...	...
220	1.0
129	1.0
139	1.0
103	1.0
467	0.0

400 rows × 1 columns

In [21]:
```
x_test
```

Out[21]:

	x_coord	y_coord
319	0.395097	0.300081
315	0.295482	0.731606
225	0.737526	0.224499
456	0.630959	0.980924
265	0.975350	0.167983
...	...	...
295	0.308642	0.733245
274	0.984383	0.220234
239	0.705457	0.204410
384	0.794578	0.507080
374	0.255745	0.802690

100 rows × 2 columns

In [22]:
```
y_test
```

Out[22]:

	veg
319	0.0
315	0.0
225	1.0
456	0.0
265	0.0
...	...
295	0.0
274	0.0
239	1.0
384	0.0
374	0.0

100 rows × 1 columns

Section 2: Decision Tree Classifiers

In [23]:
```
tree = DecisionTreeClassifier(max_depth = 5)
tree.fit(x_train, y_train)
```

Out[23]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, 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='deprecated',
                       random_state=None, splitter='best')
```

In [24]:
```
y_test_preds = tree.predict(x_test)
```

In [25]:
```
confusion_matrix(y_test , y_test_preds)
```

Out[25]:
```
array([[37, 10],
       [ 1, 52]], dtype=int64)
```

Section 3: Understanding decision trees through Visualization

In [29]:

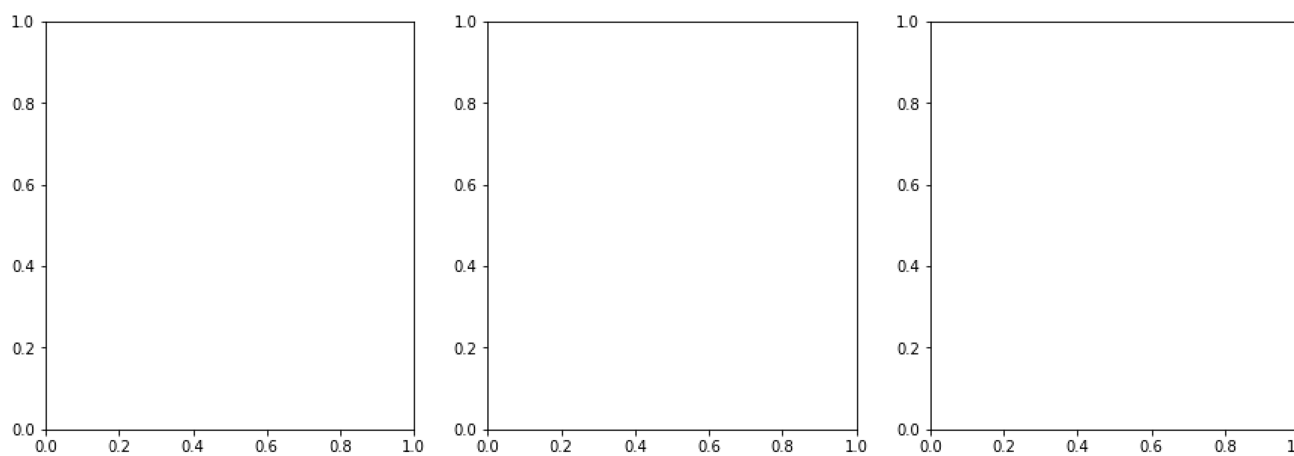
```python
def scatter_plot_data(x_df, y_series, ax):
    '''
    scatter_plot_data scatter plots the satellite data. A point in the plot is colored
    vegetation is present and 'gray' otherwise.

    input:
       x_df - a DataFrame of size N x 2, each row is a location, each column is a coord
       y_series - a Series of length N, each entry is either 0 (no vegetation) or 1 (ve
       ax - axis to plot on
    returns:
       ax - the axis with the scatter plot
    '''

    # convert x_df and y_series into numpy arrays
    x = x_df.values
    y = y_series.values

    ax.scatter(x[y == 1, 0], x[y == 1, 1], alpha=0.2, c='green', label='vegetation')
    ax.scatter(x[y == 0, 0], x[y == 0, 1], alpha=0.2, c='gray', label='nonvegetation')
    ax.set_xlim([0, 1])
    ax.set_ylim([0, 1])
    ax.set_xlabel('Latitude')
    ax.set_ylabel('Longitude')
    ax.legend(loc='best')
    return ax

def plot_decision_boundary(x_df, y_series, model, ax, plot_boundary_only=False):
    '''
    plot_decision_boundary plots the training data and the decision boundary of the cla
    input:
       x_df - a DataFrame of size N x 2, each row is a location, each column is a coord
       y_series - a Series of length N, each entry is either 0 (non-vegetation) or 1 (v
       model - the 'sklearn' classification model
       ax - axis to plot on
       poly_degree - the degree of polynomial features used to fit the model
    returns:
       ax - the axis with the scatter plot

    '''

    # convert x_df and y_series into numpy arrays
    x = x_df.values
    y = y_series.values

    # Plot data
    if not plot_boundary_only:
        ax.scatter(x[y == 1, 0], x[y == 1, 1], alpha=0.2, c='green', label='vegetation'
        ax.scatter(x[y == 0, 0], x[y == 0, 1], alpha=0.2, c='gray', label='non-vegetati

    # Create mesh
    interval = np.arange(0,1,0.01)
    n = np.size(interval)
    x1, x2 = np.meshgrid(interval, interval)
    x1 = x1.reshape(-1, 1)
    x2 = x2.reshape(-1, 1)
    xx = np.concatenate((x1, x2), axis=1)

    # Predict on mesh points
    yy = model.predict(xx)
    yy = yy.reshape((n, n))
```

```python
    # Plot decision surface
    x1 = x1.reshape(n, n)
    x2 = x2.reshape(n, n)
    if not plot_boundary_only:
        ax.contourf(x1, x2, yy, alpha=0.1, cmap='Greens')
    ax.contour(x1, x2, yy, colors='black', linewidths=0.1)
    ax.set_xlim([0, 1])
    ax.set_ylim([0, 1])
    ax.set_xlabel('Latitude')
    ax.set_ylabel('Longitude')
    ax.legend(loc='best')
    return ax
```

In [30]:

```python
# set up to create two plots in the same image
fig, ax = plt.subplots(1, 3, figsize=(15, 5))

# visualize the data on the first plot (ax[0])
scatter_plot_data(x_train, y_train, ax[0])
ax[0].set_title('Training Data')

#  plot the training data and decision tree boundary on the second plot (ax[1])
plot_decision_boundary(x_train, y_train, tree, ax[1])
ax[1].set_title('Decision Boundary on the Training Data')

#  plot the test data and decision tree boundary on the third plot (ax[2])
plot_decision_boundary(x_test, y_test, tree, ax[2])
ax[2].set_title('Decision Boundary on the Test Data')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-30-51049b1cd664> in <module>
      3 
      4 # visualize the data on the first plot (ax[0])
----> 5 scatter_plot_data(x_train, y_train, ax[0])
      6 ax[0].set_title('Training Data')
      7 

<ipython-input-29-0719bb156aa2> in scatter_plot_data(x_df, y_series, ax)
     16     y = y_series.values
     17 
---> 18     ax.scatter(x[y == 1, 0], x[y == 1, 1], alpha=0.2, c='green', label='veget
ation')
     19     ax.scatter(x[y == 0, 0], x[y == 0, 1], alpha=0.2, c='gray', label='nonveg
etation')
     20     ax.set_xlim([0, 1])

IndexError: too many indices for array
```



In [ ]: