blair_witch_project

March 9, 2021

1 W207 Final Project

1.0.1 Team: Blair Witch Project

- Kevin Ngo
- Sharon Wu
- Vish Pillai
- Blair Jones

1.0.2 Project: Forest Cover Type Prediction

The challenge To predict seven different forest cover types in four different wilderness areas of the Roosevelt National Forest of Northern Colorado with the best accuracy.

These areas represent forests with minimal human-caused disturbances, so that existing forest cover types are more a result of ecological processes rather than forest management practices.

The actual forest cover type for a given 30 x 30 meter cell was determined from US Forest Service data. Independent variables were then derived from data obtained from the US Geological Survey and USFS.

The data contains binary columns of data for qualitative independent variables such as wilderness areas and soil type.

An overview of the 4 wilderness areas

```
- Rawah Wilderness Area
- Neota Wilderness Area
- Comanche Peak Wilderness Area
- Cache la Poudre Wilderness Area
```

2 EDA

[1]:
```
#!pip install seaborn  # uncomment if not already installed
#!pip install keras # uncomment if not already installed
#!pip install tensorflow # uncomment if not already installed
```

[2]:
```
%matplotlib inline
import math
import time
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.decomposition import PCA, SparsePCA, TruncatedSVD
from sklearn.ensemble import *
from sklearn.preprocessing import normalize
import math
from sklearn.model_selection import train_test_split
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier
from sklearn import model_selection
from sklearn.metrics import accuracy_score, plot_confusion_matrix,␣
↪confusion_matrix as cm
import seaborn as sn
import keras
from keras.models import Sequential
from keras.layers import Dense
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils.np_utils import to_categorical
from itertools import combinations
```

[3]:
```
df_train = pd.read_csv('train.csv')
df_train.sample(5)
```

[3]:
```
        Id  Elevation  Aspect  Slope  Horizontal_Distance_To_Hydrology  \
3866  3867       2133     233      9                                30
8216  8217       3132      52      9                                60
6939  6940       2610     110     14                               510
6103  6104       2311     177     20                                42
4669  4670       2218     303     30                               180

      Vertical_Distance_To_Hydrology  Horizontal_Distance_To_Roadways  \
3866                               0                             1235
8216                              -1                             1892
6939                              33                              649
6103                              13                             1260
4669                             129                              671

      Hillshade_9am  Hillshade_Noon  Hillshade_3pm  …  Soil_Type32  \
3866            205             249            181  …            0
8216            225             221            130  …            0
6939            243             224            107  …            0
6103            226             247            144  …            0
4669            126             211            218  …            0
```

```
      Soil_Type33  Soil_Type34  Soil_Type35  Soil_Type36  Soil_Type37  \
3866            0            0            0            0            0
8216            0            0            0            0            0
6939            0            0            0            0            0
6103            0            0            0            0            0
4669            0            0            0            0            0

      Soil_Type38  Soil_Type39  Soil_Type40  Cover_Type
3866            0            0            0           4
8216            0            0            0           1
6939            0            0            0           5
6103            0            0            0           4
4669            0            0            0           6

[5 rows x 56 columns]
```

[4]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15120 entries, 0 to 15119
Data columns (total 56 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Id                                  15120 non-null  int64
 1   Elevation                           15120 non-null  int64
 2   Aspect                              15120 non-null  int64
 3   Slope                               15120 non-null  int64
 4   Horizontal_Distance_To_Hydrology    15120 non-null  int64
 5   Vertical_Distance_To_Hydrology      15120 non-null  int64
 6   Horizontal_Distance_To_Roadways     15120 non-null  int64
 7   Hillshade_9am                       15120 non-null  int64
 8   Hillshade_Noon                      15120 non-null  int64
 9   Hillshade_3pm                       15120 non-null  int64
 10  Horizontal_Distance_To_Fire_Points  15120 non-null  int64
 11  Wilderness_Area1                    15120 non-null  int64
 12  Wilderness_Area2                    15120 non-null  int64
 13  Wilderness_Area3                    15120 non-null  int64
 14  Wilderness_Area4                    15120 non-null  int64
 15  Soil_Type1                          15120 non-null  int64
 16  Soil_Type2                          15120 non-null  int64
 17  Soil_Type3                          15120 non-null  int64
 18  Soil_Type4                          15120 non-null  int64
 19  Soil_Type5                          15120 non-null  int64
 20  Soil_Type6                          15120 non-null  int64
 21  Soil_Type7                          15120 non-null  int64
 22  Soil_Type8                          15120 non-null  int64
```

```
 23   Soil_Type9                            15120 non-null  int64
 24   Soil_Type10                           15120 non-null  int64
 25   Soil_Type11                           15120 non-null  int64
 26   Soil_Type12                           15120 non-null  int64
 27   Soil_Type13                           15120 non-null  int64
 28   Soil_Type14                           15120 non-null  int64
 29   Soil_Type15                           15120 non-null  int64
 30   Soil_Type16                           15120 non-null  int64
 31   Soil_Type17                           15120 non-null  int64
 32   Soil_Type18                           15120 non-null  int64
 33   Soil_Type19                           15120 non-null  int64
 34   Soil_Type20                           15120 non-null  int64
 35   Soil_Type21                           15120 non-null  int64
 36   Soil_Type22                           15120 non-null  int64
 37   Soil_Type23                           15120 non-null  int64
 38   Soil_Type24                           15120 non-null  int64
 39   Soil_Type25                           15120 non-null  int64
 40   Soil_Type26                           15120 non-null  int64
 41   Soil_Type27                           15120 non-null  int64
 42   Soil_Type28                           15120 non-null  int64
 43   Soil_Type29                           15120 non-null  int64
 44   Soil_Type30                           15120 non-null  int64
 45   Soil_Type31                           15120 non-null  int64
 46   Soil_Type32                           15120 non-null  int64
 47   Soil_Type33                           15120 non-null  int64
 48   Soil_Type34                           15120 non-null  int64
 49   Soil_Type35                           15120 non-null  int64
 50   Soil_Type36                           15120 non-null  int64
 51   Soil_Type37                           15120 non-null  int64
 52   Soil_Type38                           15120 non-null  int64
 53   Soil_Type39                           15120 non-null  int64
 54   Soil_Type40                           15120 non-null  int64
 55   Cover_Type                            15120 non-null  int64
dtypes: int64(56)
memory usage: 6.5 MB
```

[5]:
```
df_train_pt = df_train.pivot_table(df_train.columns,
                ['Cover_Type'], aggfunc='mean')
df_train_pt
```

[5]:
```
                Aspect    Elevation  Hillshade_3pm  Hillshade_9am  \
Cover_Type
1           159.463426  3128.025926     144.065741     211.690278
2           151.097222  2922.540278     142.950926     214.044444
3           173.672685  2398.423148     141.549537     201.655556
4           138.099537  2223.420370     111.808796     227.968056
5           137.992130  2786.801389     121.392593     223.368981
```

```
6           180.617130  2423.276852     147.682407     193.562963
7           155.794444  3362.769907     136.193981     216.639815

            Hillshade_Noon  Horizontal_Distance_To_Fire_Points  \
Cover_Type
1               223.248611                         1994.412963
2               225.369907                         2155.277315
3               216.561111                          916.909722
4               216.889815                          860.540741
5               218.317130                         1530.388889
6               209.960648                         1057.654167
7               222.412037                         2062.847222

            Horizontal_Distance_To_Hydrology  Horizontal_Distance_To_Roadways  \
Cover_Type
1                                 271.507407                      2579.715741
2                                 287.728704                      2425.791667
3                                 210.723148                       969.595833
4                                 104.537500                       915.100463
5                                 208.873148                      1329.318519
6                                 160.095370                      1064.980556
7                                 346.904630                      2713.659722

                     Id      Slope  …  Soil_Type5  Soil_Type6  Soil_Type7  \
Cover_Type                          …
1           7996.077778  13.112963  …    0.000000    0.000000           0
2           6312.696759  13.423611  …    0.000000    0.003241           0
3           8127.537500  20.628704  …    0.025463    0.114815           0
4           6354.585648  18.468519  …    0.018056    0.112963           0
5           6486.800463  16.724537  …    0.000000    0.000000           0
6           8061.305093  18.986111  …    0.032870    0.069907           0
7           9584.496759  14.166667  …    0.000000    0.000000           0

            Soil_Type8  Soil_Type9  Vertical_Distance_To_Hydrology  \
Cover_Type
1             0.000000    0.000463                       41.281481
2             0.000463    0.004167                       47.337963
3             0.000000    0.000000                       64.081944
4             0.000000    0.000000                       40.143519
5             0.000000    0.000000                       50.871296
6             0.000000    0.000000                       44.873611
7             0.000000    0.000000                       68.945833

            Wilderness_Area1  Wilderness_Area2  Wilderness_Area3  \
Cover_Type
1                   0.491667          0.083796          0.424537
2                   0.525000          0.030556          0.435185
```

```
3                    0.000000           0.000000           0.399537
4                    0.000000           0.000000           0.000000
5                    0.396296           0.000000           0.603704
6                    0.000000           0.000000           0.445370
7                    0.252315           0.116667           0.631019

            Wilderness_Area4
Cover_Type
1                   0.000000
2                   0.009259
3                   0.600463
4                   1.000000
5                   0.000000
6                   0.554630
7                   0.000000

[7 rows x 55 columns]
```

In the data we can see that: - training dataset has 15120 entries and 56 columns - there are no missing values - there do not appear to be any duplicates - data values are unscaled - Wilderness_Area and Soil_Type columns have binary values (0,1) - each row is assigned to one distinct Wilderness Area column - each row is assigned to one distinct Soil Type column - Vertical_Distance_To_Hydrology contains negative values (ex. if the location is at a lower elevation than the water source) - the data is evenly distributed between each Cover Type, with 2,160 rows each

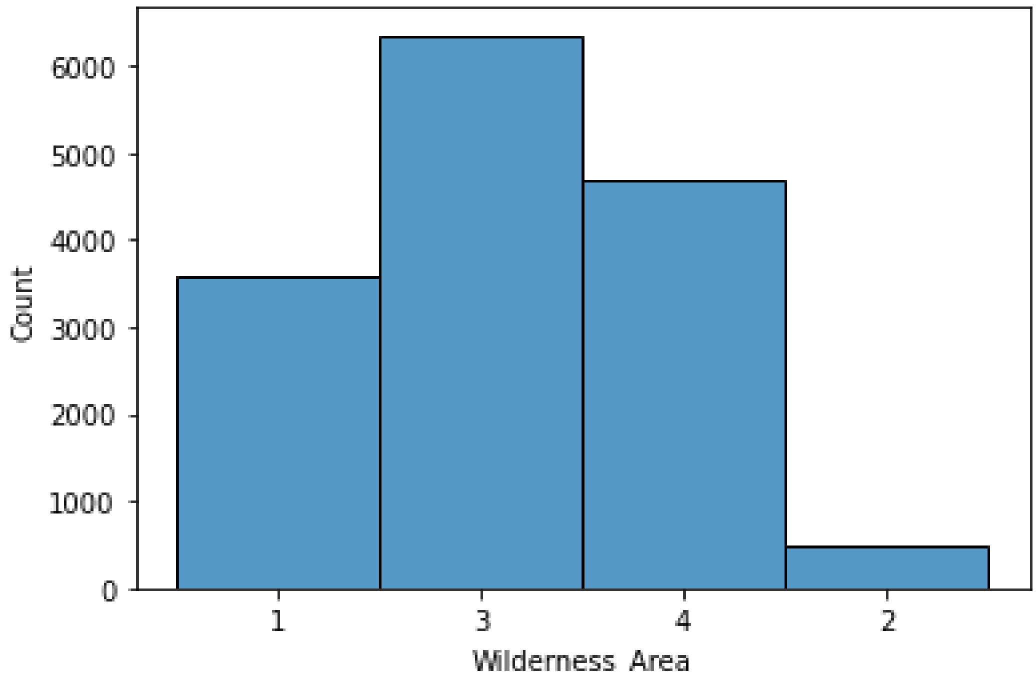
Cover_Type is our target column.

[6]:
```
# This creates a dataset for EDA manipulation
df_eda = df_train.copy()

# This creates 2 new columns that summarize the Wilderness Area and Soil Type␣
↪columns for ease of visualization
df_eda['Wilderness_Area'] = df_eda.iloc[:,11:15].idxmax(axis=1).str.
↪replace('Wilderness_Area','')
df_eda['Soil_Type'] = df_eda.iloc[:,16:55].idxmax(axis=1).str.
↪replace('Soil_Type','')
```

As seen in the next plot, the distribution is skewed across Wilderness Area.

[8]:
```
sn.histplot(data=df_eda, x='Wilderness_Area')
```

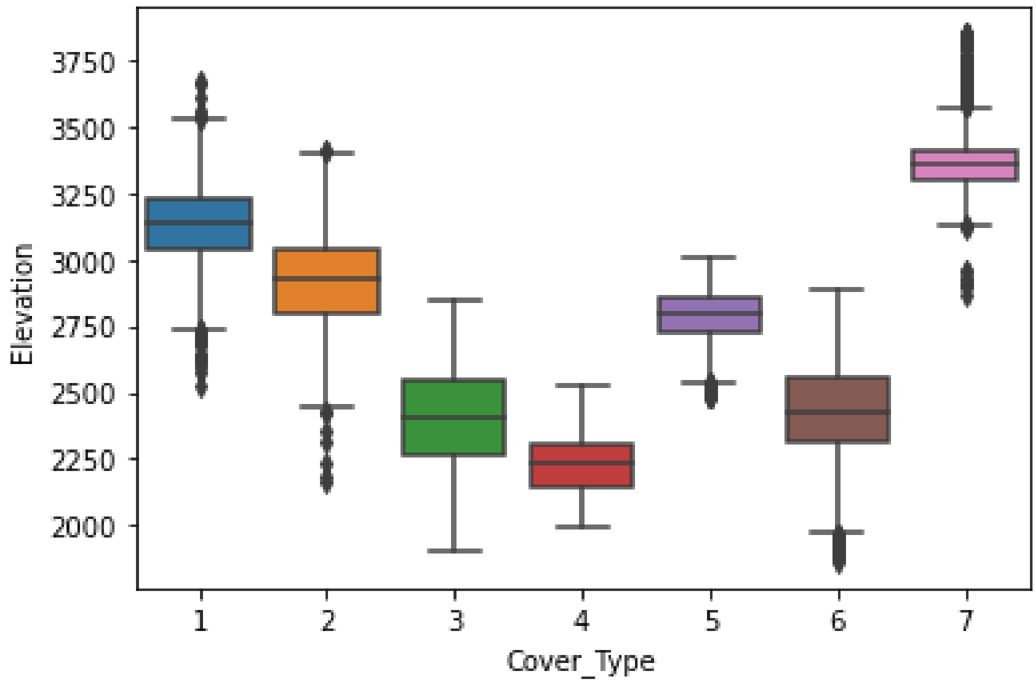
[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd8637c9f50>
```



We see that Elevation varies significantly for the different types of Cover and across Wilderness Areas.

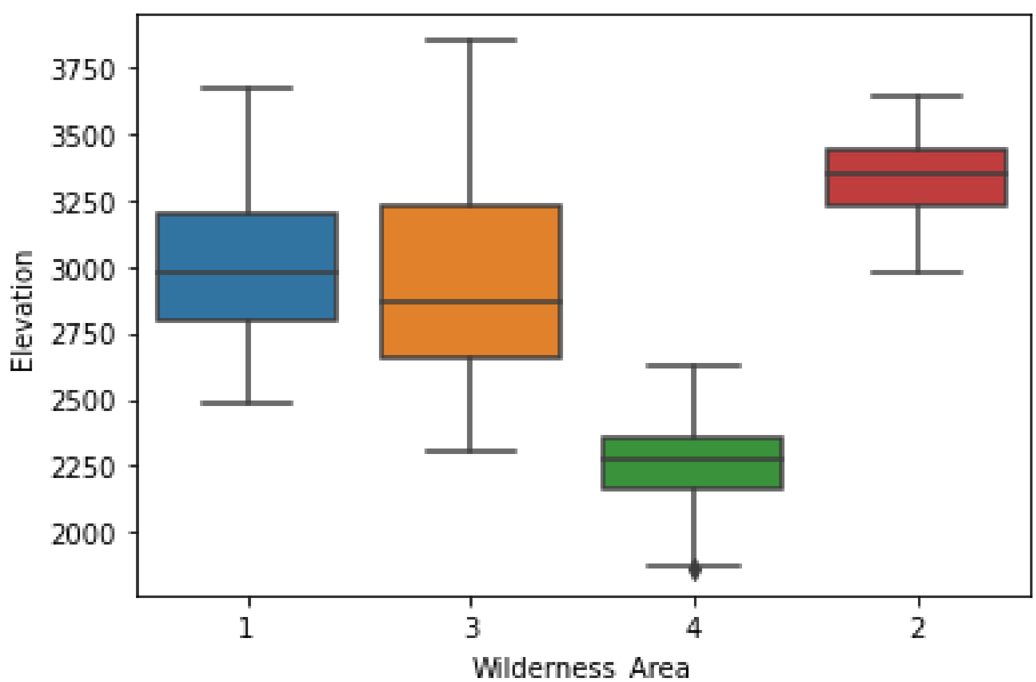
[9]:
```
sn.boxplot(data=df_eda, x='Cover_Type', y='Elevation')
```

[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd862cdb050>
```



[10]:
```
sn.boxplot(data=df_eda, x='Wilderness_Area', y='Elevation')
```

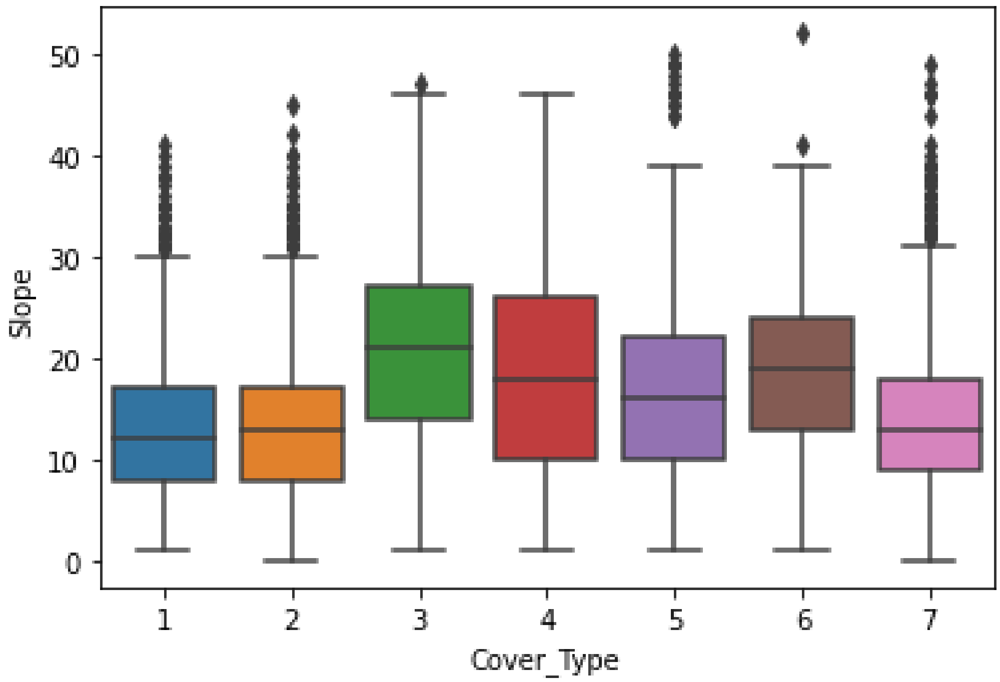
[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd865460190>



Slope and Aspect appear to be similar across the Cover Types.

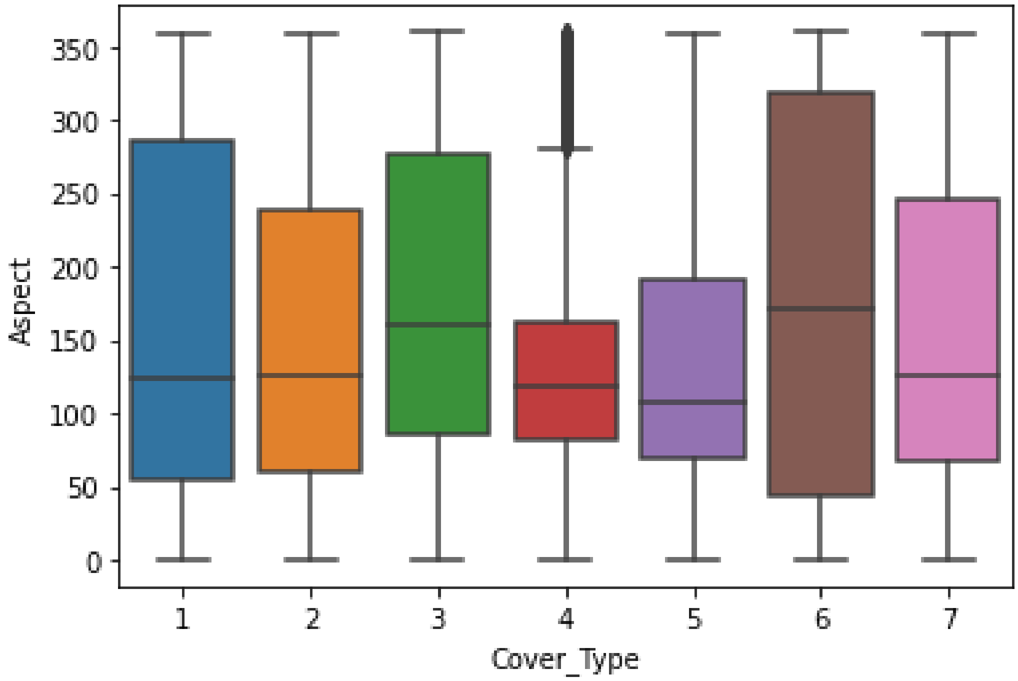
[11]:
```
sn.boxplot(data=df_eda, x='Cover_Type', y='Slope')
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd86478d850>



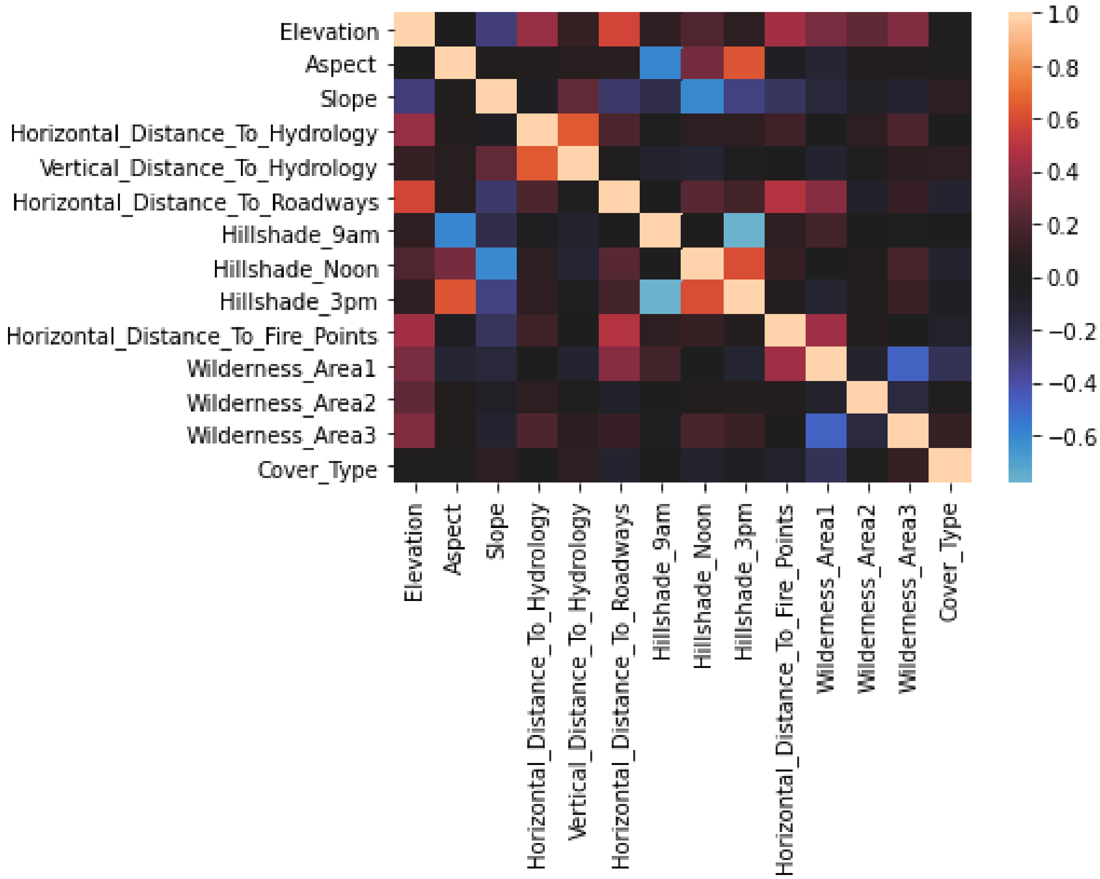
[12]:
```
sn.boxplot(data=df_eda, x='Cover_Type', y='Aspect')
```

[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd863b35ed0>



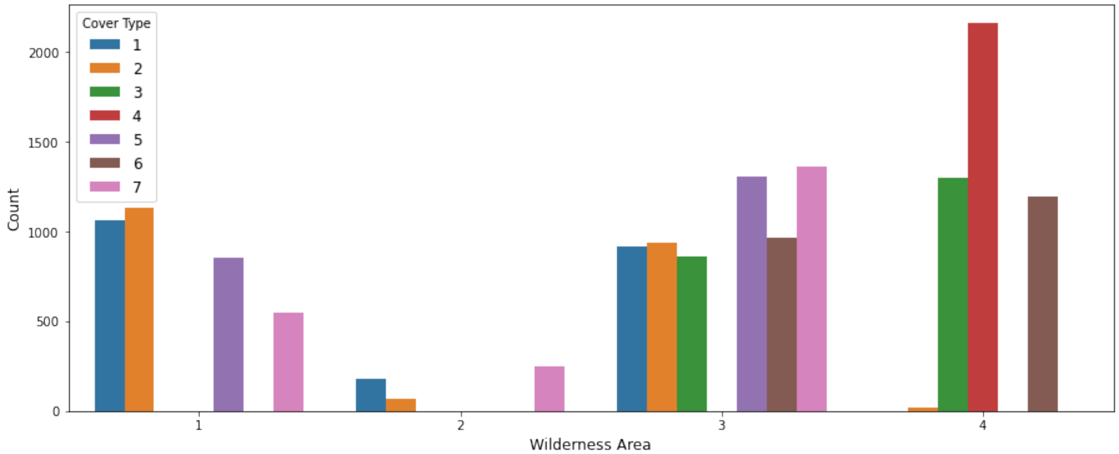

```
[13]: small_eda = df_eda.iloc[:,1:14]
      small_eda['Cover_Type'] = df_eda['Cover_Type']
      sn.heatmap(small_eda.corr(), center=0)
```

```
[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd864a3cb50>
```



2.1 Feature: Cover_Type & Wilderness_Areas

[15]:
```
plt.figure(figsize=(15,6))
order = ["1","2","3","4"]
ax = sn.countplot(x= "Wilderness_Area", hue = "Cover_Type", data = df_eda,␣
↪order=order)
ax.set_xlabel("Wilderness Area",fontsize=12)
ax.set_ylabel("Count",fontsize=12)
plt.legend(title="Cover Type", loc='upper left',fontsize=12)
plt.show()
```



Some cover types only belong specific wilderness areas, in this case wilderness areas could be a significant variables in the predictive model.

2.2 Data Transformations

[16]:
```
# normalization
from sklearn.preprocessing import normalize

col_normalize =␣
↪['Aspect','Slope','Horizontal_Distance_To_Hydrology','Vertical_Distance_To_Hydrology',␣
↪'Hillshade_9am','Hillshade_Noon','Hillshade_3pm','Horizontal_Distance_To_Fire_Points']
df_train_norm = df_train.copy()
df_train_norm[col_normalize] = normalize(df_train[col_normalize])
```

to be completed

copy from Sharon/Vish/Kevin notebooks for EDA modeling examples

```
- WITHOUT TRAINING RESULTS
- THIS IS JUST FOR PCA/Feature Selection
```

3 Hypothesis

Based on the EDA, we believe that a Decision Tree approach will yield the best classification results and also serve as the basis for a good predictor on the test dataset.

One features is strongly linked to a specific Cover Type: Cover Type 4 is only found in Wilderness Area 4. However there is significant overlap of other features for all Cover Types and Wildnerness Areas. This suggests that a strategy using smaller Decision Trees to classify sub-groupings of features will be effective.

4 Modeling

There are many models we can test, including: - Decision Tree - Random Forest - XGBoost - Neural Network - etc.

This first pass will examine RandomForest against the Training dataset, using random sampling to extract a sub-training set and then evaluating performance of the trained classifier against the remaining data in the training data.

This way we avoid touching the true Testing dataset until we are satisfied that our model performs as expected and should also generalize.

[17]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, plot_confusion_matrix,␣
↪confusion_matrix as cm
from sklearn.model_selection import train_test_split
from sklearn import model_selection
```

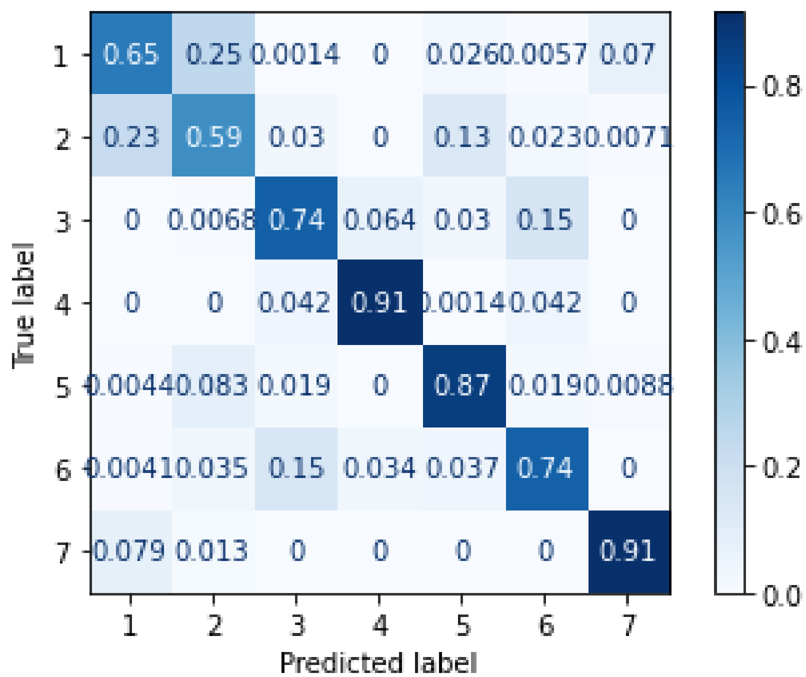
[18]:
```
np.set_printoptions(precision=5)
def score_model(model,df, return_val=False, return_train=False, display=True,␣
↪return_acc=False, return_time=False, show_weights=False):
    X , Y = df.drop(columns=['Id','Cover_Type']).to_numpy(), df.Cover_Type.
↪to_numpy()
    X_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=.33,␣
↪random_state=0)
    start = time.time()
    results = model_selection.cross_val_score(model, X, Y, cv=kfold)
    model.fit(X_train, y_train)
    pred = model.predict(X_val)
    acc = accuracy_score(y_val, pred)
    end = time.time()
    print('\nModel:',type(model).__name__)
    print('\tcv acc:', round(results.mean(),4))
    print('\tsplit acc:', round(acc,4))
    print('\ttime taken:', round(end-start, 4))
    if display:
        matrix = cm(y_val, pred)
        print('\t', matrix.diagonal() / matrix.sum(axis=1))

        disp = plot_confusion_matrix(model, X_val, y_val,␣
↪display_labels=set(y_train), cmap=plt.cm.Blues, normalize='true')
        plt.show()

    if show_weights:
        for w,k in sorted(list(zip(model.feature_importances_, df.
↪drop(columns=['Id','Cover_Type']).columns)), key=lambda x: x[0]):
```

```
            print(k,w)

    # return all data
    return_data = [model]
    if return_train:
        return_data += [X_train, y_train]
    if return_val:
        return_data += [X_val, y_val]
    if return_acc:
        return_data += [acc]
    if return_time:
        return_data += [end-start]
    return tuple(return_data)
```

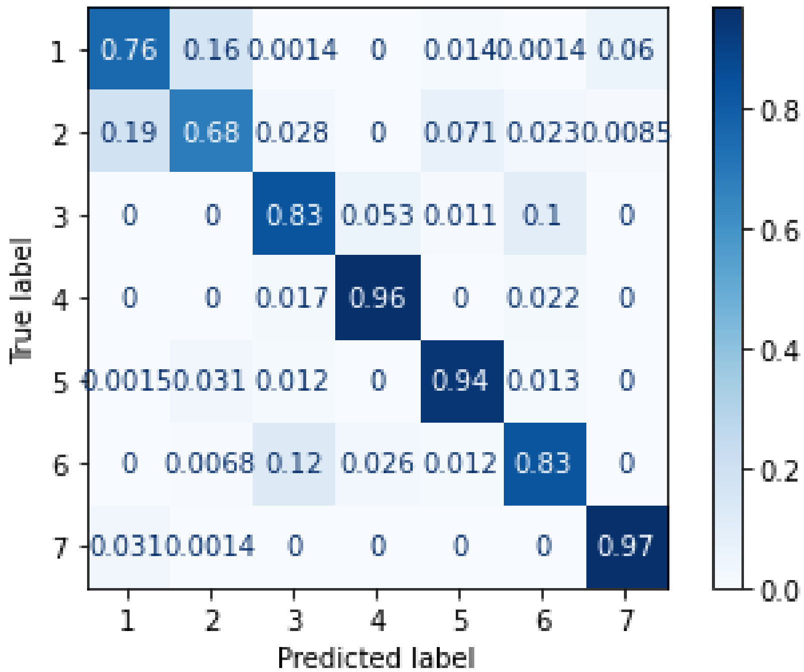
[19]:
```
kfold = model_selection.KFold(n_splits=10, random_state=0, shuffle=True)
```

[22]:
```
models = []
for clf in [
    DecisionTreeClassifier(max_depth=14, random_state=0), # 14 was determine by␣
 ↪iterating
    RandomForestClassifier(n_jobs=-1, random_state=0),]:

    models.append(score_model(clf,df_train_norm))
```

```
Model: DecisionTreeClassifier
        cv acc: 0.7877
        split acc: 0.7727
        time taken: 1.4429
         [0.65199 0.58582 0.74491 0.91457 0.8653  0.73978 0.90884]
```




```
Model: RandomForestClassifier
        cv acc: 0.8743
        split acc: 0.8539
        time taken: 8.7009
         [0.75852 0.68369 0.83446 0.96078 0.9429  0.83106 0.96774]
```

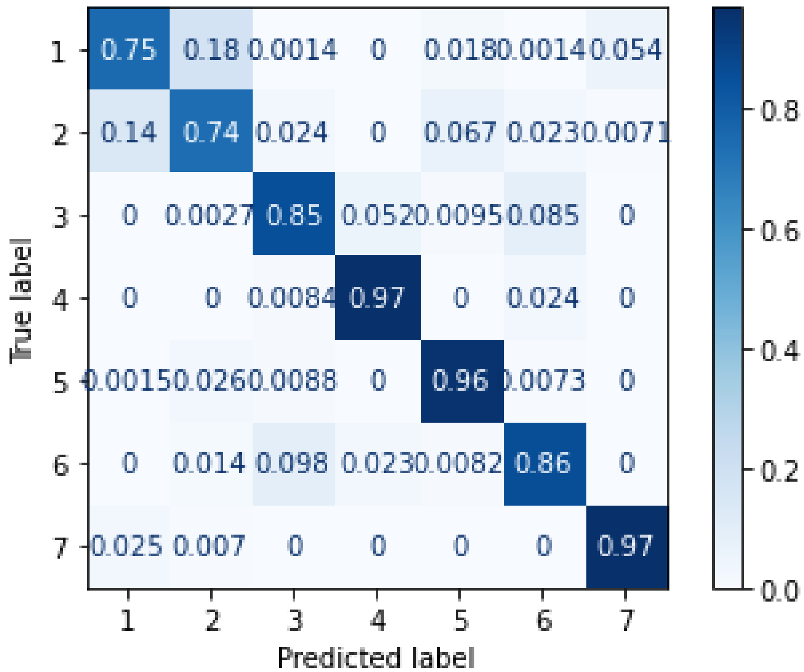


RandomForest performs much better than the DecisionTree Classifier.

Next we will examine the ExtraTreesClassifier.

[24]:
```
# Determining RF model - ExtraTreesClassifier

models.append(score_model(ExtraTreesClassifier(n_jobs=-1,␣
↪random_state=0),df_train_norm))
```

```
Model: ExtraTreesClassifier
        cv acc: 0.8849
        split acc: 0.8693
        time taken: 8.9207
         [0.74858 0.73901 0.85075 0.96779 0.95608 0.85695 0.96774]
```



The ExtraTrees Classifier improved upon the performance of RandomForest for several categories.

5 Current Status

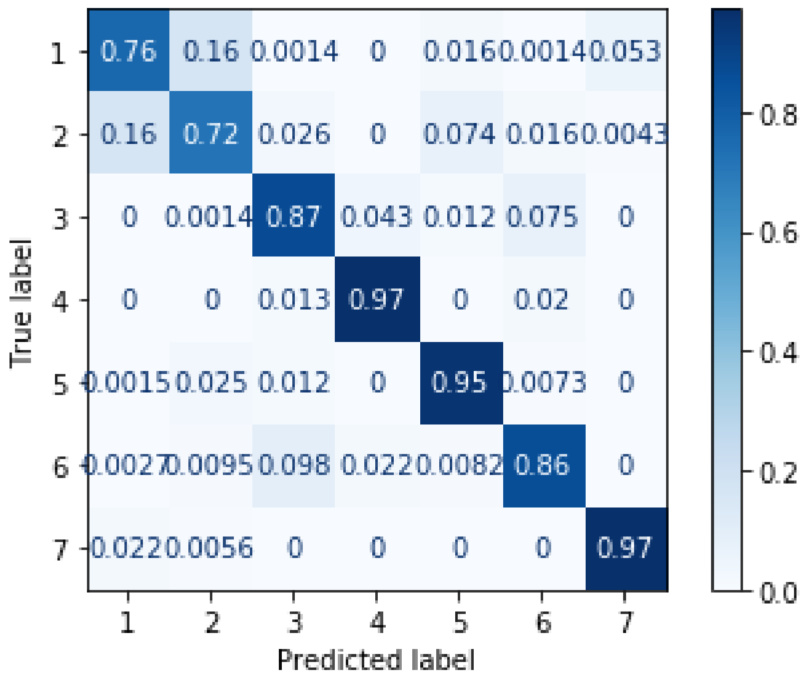
After normalization, we can see the weaknesses of our 'ExtraTrees' model is the cluster involving Cover Type '1' and '2'. We will focus now on optimizing our model to better classify this smaller cluster and to get our cv accuracy at 90%+.

work in progress

[40]:
```
col_normalize =␣
↪['Aspect','Slope','Horizontal_Distance_To_Hydrology','Vertical_Distance_To_Hydrology',
                ␣
↪'Hillshade_9am','Hillshade_Noon','Hillshade_3pm','Horizontal_Distance_To_Fire_Points',
                 'Horizontal_Distance_To_Roadways']
df_train_norm = df_train.copy()
df_train_norm[col_normalize] = normalize(df_train[col_normalize])
df_train_norm['log_Horizontal_Distance_To_Roadways'] =␣
↪(df_train['Horizontal_Distance_To_Roadways']+1).apply(np.log)
best_m, best_X_train, best_y_train, best_X_val, best_y_val =␣
↪score_model(ExtraTreesClassifier(n_jobs=-1, random_state=0), df_train_norm,␣
↪return_train = True, return_val = True)
```

```
Model: ExtraTreesClassifier
```

```
cv acc: 0.8901
split acc: 0.8723
time taken: 15.2719
 [0.7642  0.72057 0.86839 0.96779 0.95461 0.85967 0.97195]
```




```
[ ]:
```