STAT 6500

Statistical Machine Learning

Land Use Cover - EDA

Kendall Byrd - Atitarn Dechasuravanit - Alexys Rodriguez

Spring 2022
Wednesday, March 2

kjhasd hfkj ahskjd hfkjahs dkjfh kjasdhkjf haskjdfkjas hfdkjhsa kjdfh kjsad hfkjsab qwrqwerwqer wqer ewqrq werqwerwqerwqerqwerqwerqwerq werwqerqwe rewqr

asdfakjsdhfkjahskjdf hkjsadh fkjhsadkjfh kjdsah fkjhdsa kjfh kjdsahf kjsahd kjfh dsakfh

ahskjdfh asdkj hfkjsadh fkjdsa

```
#load in training and test data
train = pd.read_csv('training.csv')
test = pd.read_csv('testing.csv')
print("Rows and Columns(Train): ",train.shape)
```

```
## Rows and Columns(Train):  (168, 148)
```

```
print("Rows and Columns(Test) : ",test.shape)
```

```
## Rows and Columns(Test) :  (507, 148)
```

```
# check for missing values although it is clear there are none
train.isnull().any().any()
```

```
## False
```

```
# duplicate function of pandas returns a duplicate row as true and others as false
sum(train.duplicated())
```

```
## 0
```

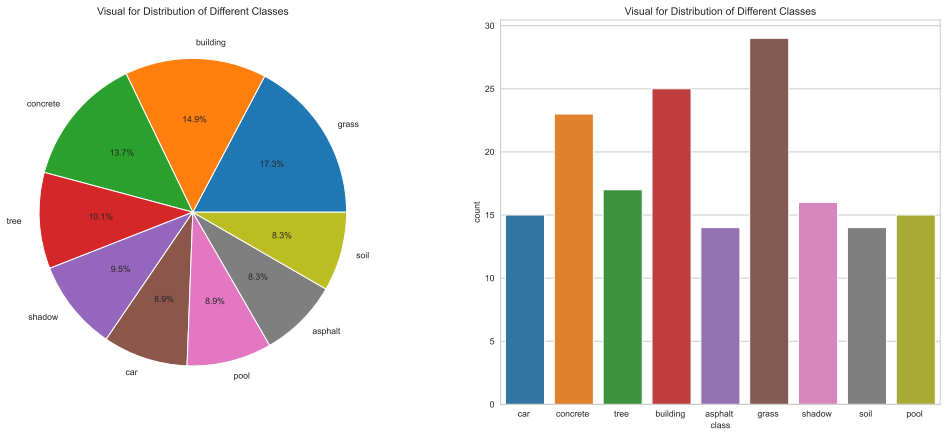
```
# basic statistical details
fig = train.describe().T
fig = fig.round(5)  # round to 5 decimal places
table = go.Table(
    columnwidth=[0.8]+[0.5]*8,
    header=dict(
        values=['Attribute'] + list(fig.columns),
        line = dict(color='darkslategray'),
        fill = dict(color='royalblue'),
    ),
    cells=dict(
        values=[fig.index] + [fig[k].tolist() for k in fig.columns[:]],
        line = dict(color='darkslategray'),
        fill = dict(color=['paleturquoise', 'white'])
    )
)
plot([table], filename='table-of-data.html')
```

```
## 'table-of-data.html'
```

```
# more general data exploration
print(train['class'].value_counts())
```

```
## grass        29
## building     25
## concrete     23
## tree         17
## shadow       16
## car          15
## pool         15
## asphalt      14
## soil         14
## Name: class, dtype: int64
```

```
f,axes=plt.subplots(1,2,figsize=(20,8))
train['class'].value_counts().plot.pie(autopct='%1.1f%%',ax=axes[0])
axes[0].set_title('Visual for Distribution of Different Classes')
axes[0].set_ylabel('')
sns.countplot('class',data=train,ax=axes[1]) # sns.countplot is used
                                             # like a histogram but for
                                             # categorical data
axes[1].set_title('Visual for Distribution of Different Classes')
plt.show()
```

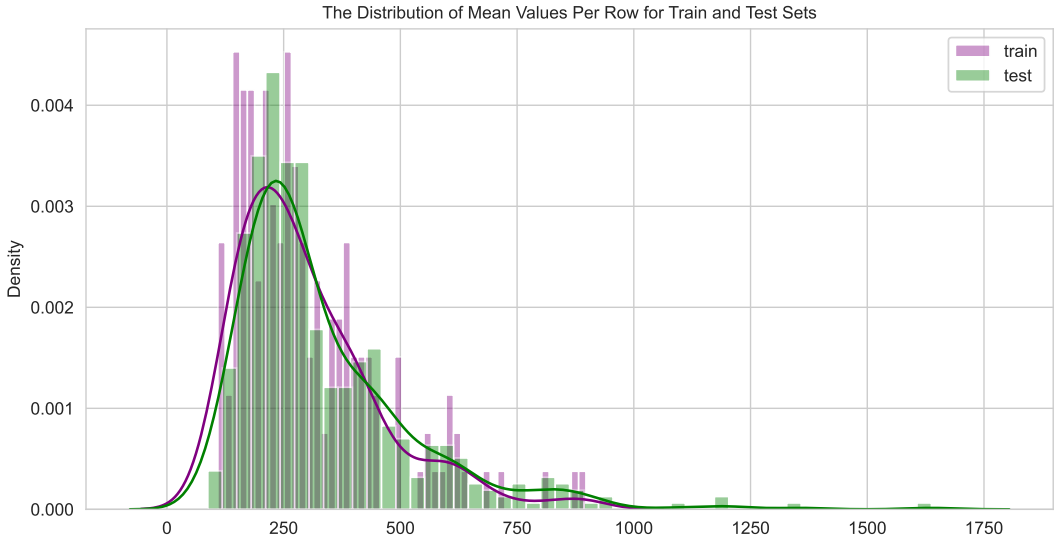



```
# Lets take a look at any outliers that could be potential issues
from collections import Counter
def examine_outliers(train_data, n, features):
  outlier_indicator = []
  for out in features:
    Q1 = np.percentile(train_data[out], 25)
    Q3 = np.percentile(train_data[out], 75)
    IQR = Q3 - Q1
    outlier_step = 1.5 * IQR # IQR method of dealing with outliers, 1 of 2 methods
    outlier_list_out = train_data[
            (train_data[out] < Q1 - outlier_step) | (train_data[out] > Q3 +
            outlier_step)].index
    outlier_indicator.extend(outlier_list_out)
    outlier_indices = Counter(outlier_indicator)
    multiple_outliers = list(k for k, j in outlier_indices.items() if j > n)
    return multiple_outliers
```

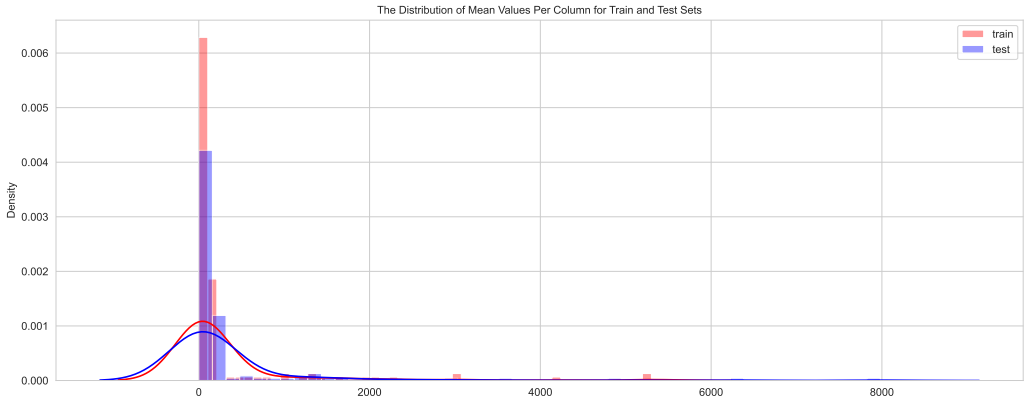
```
# find outliers that should be removed
list_atributes = train.drop('class', axis=1).columns
outliers_to_remove = examine_outliers(train, 2, list_atributes)
len(outliers_to_remove)
#outliers_to_remove
#train.loc[outliers_to_remove]
```

```
## 0
```

```
# lets look at mean values per row and column for train test data
# mean for rows
plt.figure(figsize=(10,5))
features = train.columns.values[1:148]
plt.title("The Distribution of Mean Values Per Row for Train and Test Sets",
          fontsize=10)
sns.distplot(train[features].mean(axis=1),color="purple", kde=True,bins=50,
             label='train') # kde is kernel density estimation
sns.distplot(test[features].mean(axis=1),color="green", kde=True,bins=50,
             label='test')
plt.legend()
plt.show()
```



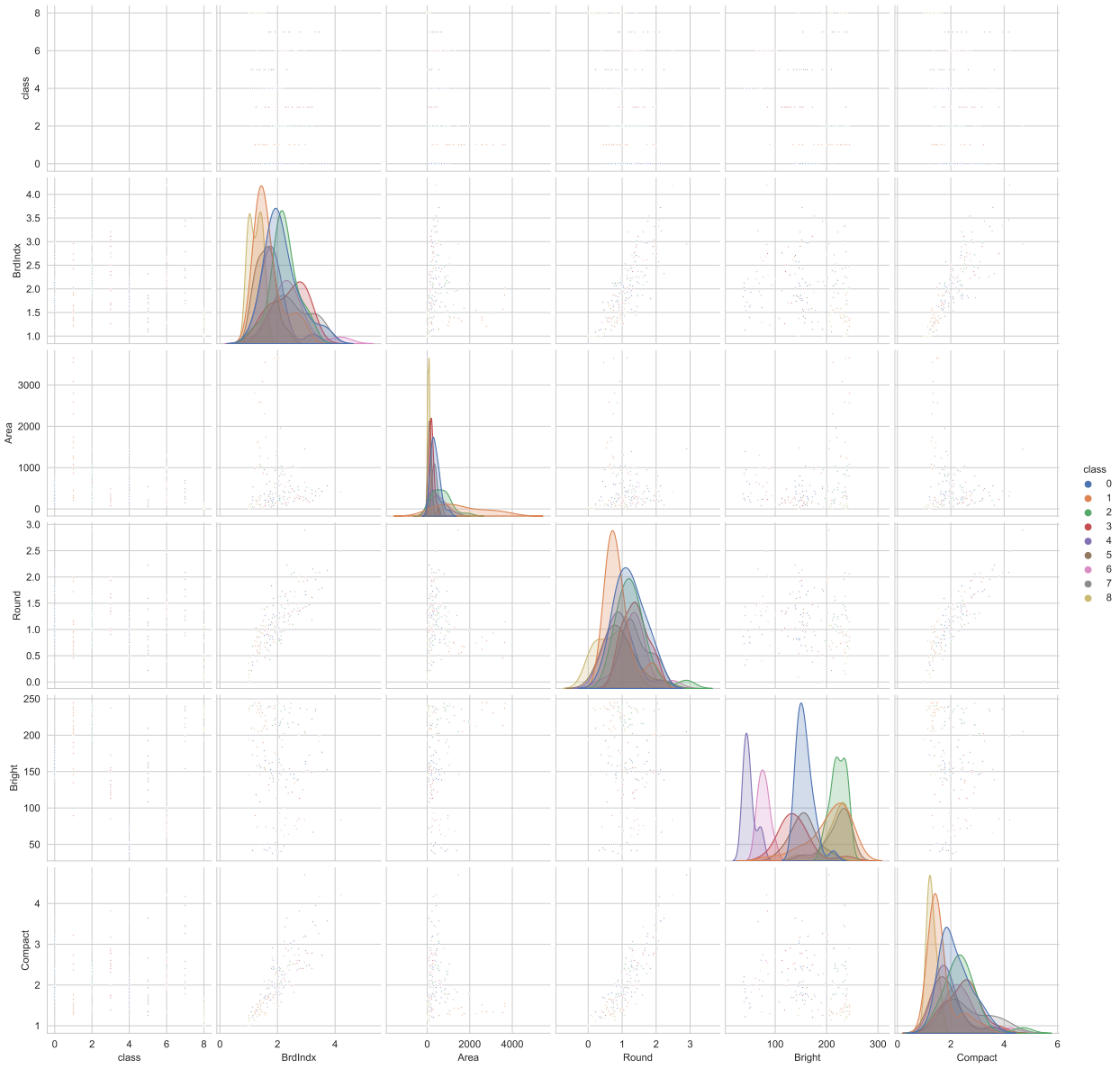

```
# mean for columns
plt.figure(figsize=(16,6))
plt.title("The Distribution of Mean Values Per Column for Train and Test Sets",
          fontsize=10)
sns.distplot(train[features].mean(axis=0),color="red",kde=True,bins=50,
             label='train')
sns.distplot(test[features].mean(axis=0),color="blue", kde=True,bins=50,
             label='test')
plt.legend()
plt.show()
```



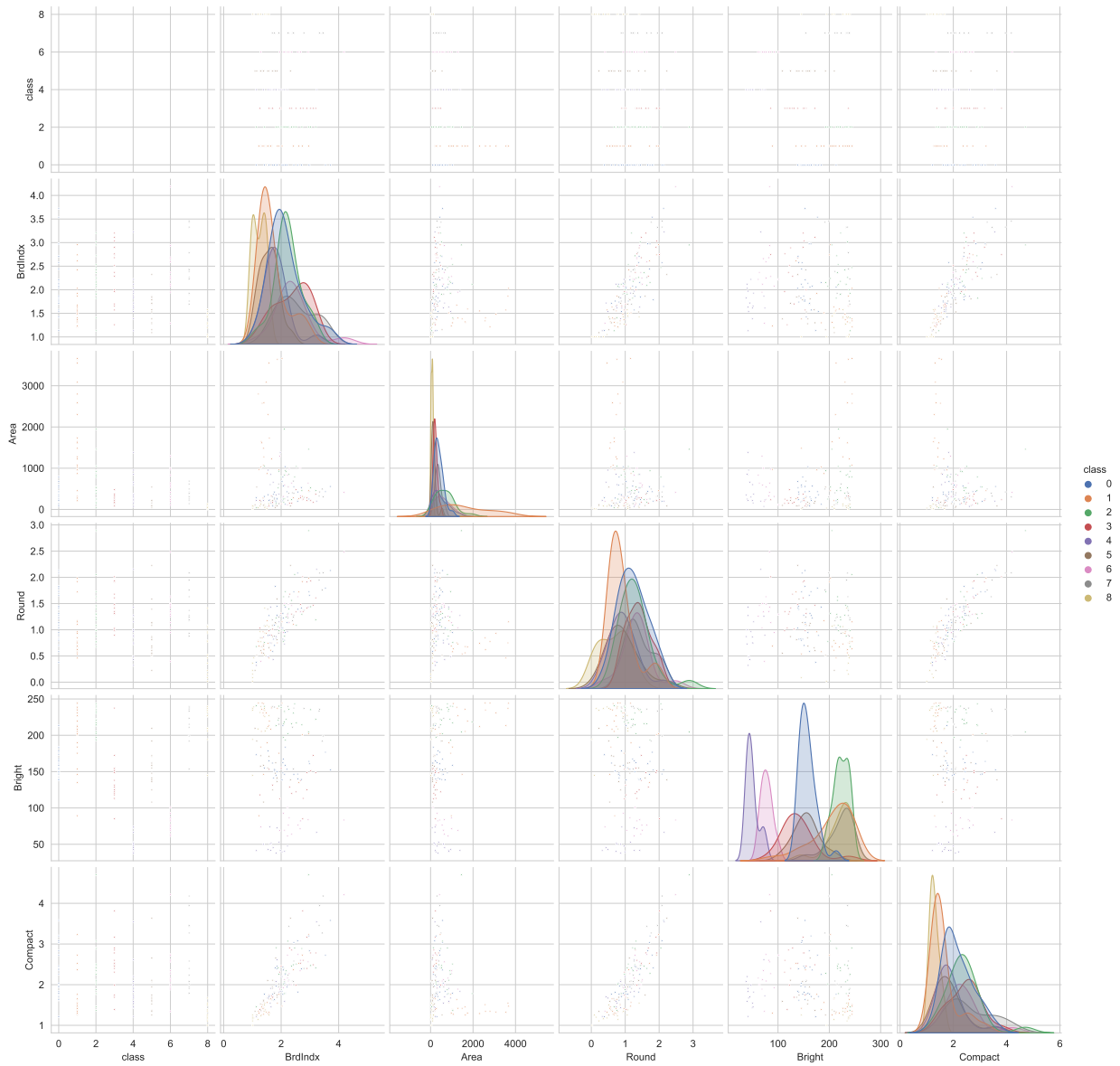

```
# lets examine correlations between features
# first the categorical variable 'class' needs to be changed to numerical variable so...
group_map = {"grass ":0,"building ":1,'concrete ':2,'tree ':3,'shadow ':4,'pool ':5,
             'asphalt ':6,'soil ':7,'car ':8}
train['class'] = train['class'].map(group_map)
test['class'] = test['class'].map(group_map)
train['class'].unique()
```

```
## array([8, 2, 3, 1, 6, 0, 4, 7, 5], dtype=int64)
```

```
# now lets look at the correlation between a few variables
sns.pairplot(train, vars=['class', 'BrdIndx','Area','Round','Bright','Compact'],
             hue='class', palette='deep', plot_kws={"s": 3})
```




```
plt.show()
```




```
# correlation of features with target
corr = train.corr().abs().unstack().sort_values(kind="quicksort").reset_index()
corr = corr[corr['level_0'] != corr['level_1']]
corr.head()
```

```
##       level_0    level_1         0
## 0        SD_R   GLCM1_60  0.000052
## 1   GLCM1_60        SD_R  0.000052
## 2   Mean_NIR  BordLngth  0.000162
## 3  BordLngth   Mean_NIR  0.000162
## 4  Mean_R_40   Round_40  0.000190
```

```
correlations = corr.loc[corr[0] == 1]
features_to_be_removed = set(list(correlations['level_1']))
correlations.shape
```

```
## (42, 3)
```