01_ForestProject-EDA-1

August 19, 2020

1 Prediciton (Classification) the Types of Trees.

1.0.1 This Jupyter Notebook contains;

- EDA (Exploratory data analysis),
- Data preparation, transformation process.

1.0.2 What I Plan to :

- Try to understand the dataset column by column using `pandas` module.
- Do research within the scope of domain (forest, trees) knowledge on the internet to get to know the data set in the fastest way.
- If needed, I will implement cleaning, handling with outliers and missing values using `pandas`, `NumPy` and other required modules.
- For the best result in modeling, I will implement Feature Engineering process using `SQLite` local database. I will,
  - import dataset into my SQLite local database,
  - produce or transform new columns,
  - get rid of unnecassary columns,
  - make the dataset ready to model.

1.0.3 Importing required libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sqlalchemy import create_engine
import warnings
from IPython.core.pylabtools import figsize
from scipy.stats import zscore
from scipy import stats
from numpy import percentile
font_title = {'family': 'times new roman', 'color': 'darkred',
```

```
              'weight': 'bold', 'size': 14}

warnings.filterwarnings('ignore')
sns.set_style("whitegrid")

plt.rcParams['figure.dpi'] = 100
```

1.0.4 Reading dataset with pandas

[2]:
```
tree = pd.read_csv("covtype.csv")
```

[3]:
```
tree.head()
```

[3]:
```
   Elevation  Aspect  Slope  Horizontal_Distance_To_Hydrology  \
0       2596      51      3                               258
1       2590      56      2                               212
2       2804     139      9                               268
3       2785     155     18                               242
4       2595      45      2                               153

   Vertical_Distance_To_Hydrology  Horizontal_Distance_To_Roadways  \
0                               0                              510
1                              -6                              390
2                              65                             3180
3                             118                             3090
4                              -1                              391

   Hillshade_9am  Hillshade_Noon  Hillshade_3pm  \
0            221             232            148
1            220             235            151
2            234             238            135
3            238             238            122
4            220             234            150

   Horizontal_Distance_To_Fire_Points  …  Soil_Type32  Soil_Type33  \
0                                6279  …            0            0
1                                6225  …            0            0
2                                6121  …            0            0
3                                6211  …            0            0
4                                6172  …            0            0

   Soil_Type34  Soil_Type35  Soil_Type36  Soil_Type37  Soil_Type38  \
0            0            0            0            0            0
1            0            0            0            0            0
2            0            0            0            0            0
3            0            0            0            0            0
```

```
4            0            0            0            0            0

   Soil_Type39  Soil_Type40  Cover_Type
0            0            0           5
1            0            0           5
2            0            0           2
3            0            0           2
4            0            0           5

[5 rows x 55 columns]
```

1.0.5 Let's get acquianted with the DataSet.

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 581012 entries, 0 to 581011
Data columns (total 55 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Elevation                           581012 non-null  int64
 1   Aspect                              581012 non-null  int64
 2   Slope                               581012 non-null  int64
 3   Horizontal_Distance_To_Hydrology    581012 non-null  int64
 4   Vertical_Distance_To_Hydrology      581012 non-null  int64
 5   Horizontal_Distance_To_Roadways     581012 non-null  int64
 6   Hillshade_9am                       581012 non-null  int64
 7   Hillshade_Noon                      581012 non-null  int64
 8   Hillshade_3pm                       581012 non-null  int64
 9   Horizontal_Distance_To_Fire_Points  581012 non-null  int64
 10  Wilderness_Area1                    581012 non-null  int64
 11  Wilderness_Area2                    581012 non-null  int64
 12  Wilderness_Area3                    581012 non-null  int64
 13  Wilderness_Area4                    581012 non-null  int64
 14  Soil_Type1                          581012 non-null  int64
 15  Soil_Type2                          581012 non-null  int64
 16  Soil_Type3                          581012 non-null  int64
 17  Soil_Type4                          581012 non-null  int64
 18  Soil_Type5                          581012 non-null  int64
 19  Soil_Type6                          581012 non-null  int64
 20  Soil_Type7                          581012 non-null  int64
 21  Soil_Type8                          581012 non-null  int64
 22  Soil_Type9                          581012 non-null  int64
 23  Soil_Type10                         581012 non-null  int64
 24  Soil_Type11                         581012 non-null  int64
 25  Soil_Type12                         581012 non-null  int64
```

```
 26   Soil_Type13                           581012 non-null   int64
 27   Soil_Type14                           581012 non-null   int64
 28   Soil_Type15                           581012 non-null   int64
 29   Soil_Type16                           581012 non-null   int64
 30   Soil_Type17                           581012 non-null   int64
 31   Soil_Type18                           581012 non-null   int64
 32   Soil_Type19                           581012 non-null   int64
 33   Soil_Type20                           581012 non-null   int64
 34   Soil_Type21                           581012 non-null   int64
 35   Soil_Type22                           581012 non-null   int64
 36   Soil_Type23                           581012 non-null   int64
 37   Soil_Type24                           581012 non-null   int64
 38   Soil_Type25                           581012 non-null   int64
 39   Soil_Type26                           581012 non-null   int64
 40   Soil_Type27                           581012 non-null   int64
 41   Soil_Type28                           581012 non-null   int64
 42   Soil_Type29                           581012 non-null   int64
 43   Soil_Type30                           581012 non-null   int64
 44   Soil_Type31                           581012 non-null   int64
 45   Soil_Type32                           581012 non-null   int64
 46   Soil_Type33                           581012 non-null   int64
 47   Soil_Type34                           581012 non-null   int64
 48   Soil_Type35                           581012 non-null   int64
 49   Soil_Type36                           581012 non-null   int64
 50   Soil_Type37                           581012 non-null   int64
 51   Soil_Type38                           581012 non-null   int64
 52   Soil_Type39                           581012 non-null   int64
 53   Soil_Type40                           581012 non-null   int64
 54   Cover_Type                            581012 non-null   int64
dtypes: int64(55)
memory usage: 243.8 MB
```

1.0.6 Summary results :

- There are 581012 rows and 55 columns.
- All columns are `int` type of data.

[5]:
```python
# There are no missing values

tree.isnull().sum()*100/tree.shape[0]
```

[5]:
```
Elevation                             0.0
Aspect                                0.0
Slope                                 0.0
Horizontal_Distance_To_Hydrology      0.0
Vertical_Distance_To_Hydrology        0.0
Horizontal_Distance_To_Roadways       0.0
```

Hillshade_9am	0.0
Hillshade_Noon	0.0
Hillshade_3pm	0.0
Horizontal_Distance_To_Fire_Points	0.0
Wilderness_Area1	0.0
Wilderness_Area2	0.0
Wilderness_Area3	0.0
Wilderness_Area4	0.0
Soil_Type1	0.0
Soil_Type2	0.0
Soil_Type3	0.0
Soil_Type4	0.0
Soil_Type5	0.0
Soil_Type6	0.0
Soil_Type7	0.0
Soil_Type8	0.0
Soil_Type9	0.0
Soil_Type10	0.0
Soil_Type11	0.0
Soil_Type12	0.0
Soil_Type13	0.0
Soil_Type14	0.0
Soil_Type15	0.0
Soil_Type16	0.0
Soil_Type17	0.0
Soil_Type18	0.0
Soil_Type19	0.0
Soil_Type20	0.0
Soil_Type21	0.0
Soil_Type22	0.0
Soil_Type23	0.0
Soil_Type24	0.0
Soil_Type25	0.0
Soil_Type26	0.0
Soil_Type27	0.0
Soil_Type28	0.0
Soil_Type29	0.0
Soil_Type30	0.0
Soil_Type31	0.0
Soil_Type32	0.0
Soil_Type33	0.0
Soil_Type34	0.0
Soil_Type35	0.0
Soil_Type36	0.0
Soil_Type37	0.0
Soil_Type38	0.0
Soil_Type39	0.0

```
Soil_Type40                          0.0
Cover_Type                           0.0
dtype: float64
```

1.0.7 Summary results :

- There is no missing value. That is great!.

1.0.8 Overall information of numerical data

[6]:
```
tree.describe()
```

[6]:
```
           Elevation         Aspect          Slope  \
count  581012.000000  581012.000000  581012.000000
mean     2959.365301     155.656807      14.103704
std       279.984734     111.913721       7.488242
min      1859.000000       0.000000       0.000000
25%      2809.000000      58.000000       9.000000
50%      2996.000000     127.000000      13.000000
75%      3163.000000     260.000000      18.000000
max      3858.000000     360.000000      66.000000

       Horizontal_Distance_To_Hydrology  Vertical_Distance_To_Hydrology  \
count                     581012.000000                   581012.000000
mean                         269.428217                       46.418855
std                          212.549356                       58.295232
min                            0.000000                     -173.000000
25%                          108.000000                        7.000000
50%                          218.000000                       30.000000
75%                          384.000000                       69.000000
max                         1397.000000                      601.000000

       Horizontal_Distance_To_Roadways  Hillshade_9am  Hillshade_Noon  \
count                    581012.000000  581012.000000   581012.000000
mean                       2350.146611     212.146049      223.318716
std                        1559.254870      26.769889       19.768697
min                           0.000000       0.000000        0.000000
25%                        1106.000000     198.000000      213.000000
50%                        1997.000000     218.000000      226.000000
75%                        3328.000000     231.000000      237.000000
max                        7117.000000     254.000000      254.000000

       Hillshade_3pm  Horizontal_Distance_To_Fire_Points  …    Soil_Type32  \
count  581012.000000                       581012.000000  …  581012.000000
mean      142.528263                         1980.291226  …       0.090392
```

```
std        38.274529                              1324.195210  ...      0.286743
min         0.000000                                 0.000000  ...      0.000000
25%       119.000000                              1024.000000  ...      0.000000
50%       143.000000                              1710.000000  ...      0.000000
75%       168.000000                              2550.000000  ...      0.000000
max       254.000000                              7173.000000  ...      1.000000

          Soil_Type33    Soil_Type34    Soil_Type35    Soil_Type36  \
count   581012.000000  581012.000000  581012.000000  581012.000000
mean         0.077716       0.002773       0.003255       0.000205
std          0.267725       0.052584       0.056957       0.014310
min          0.000000       0.000000       0.000000       0.000000
25%          0.000000       0.000000       0.000000       0.000000
50%          0.000000       0.000000       0.000000       0.000000
75%          0.000000       0.000000       0.000000       0.000000
max          1.000000       1.000000       1.000000       1.000000

          Soil_Type37    Soil_Type38    Soil_Type39    Soil_Type40  \
count   581012.000000  581012.000000  581012.000000  581012.000000
mean         0.000513       0.026803       0.023762       0.015060
std          0.022641       0.161508       0.152307       0.121791
min          0.000000       0.000000       0.000000       0.000000
25%          0.000000       0.000000       0.000000       0.000000
50%          0.000000       0.000000       0.000000       0.000000
75%          0.000000       0.000000       0.000000       0.000000
max          1.000000       1.000000       1.000000       1.000000

           Cover_Type
count   581012.000000
mean         2.051471
std          1.396504
min          1.000000
25%          1.000000
50%          2.000000
75%          2.000000
max          7.000000

[8 rows x 55 columns]
```

1.0.9 Number of Unique values of each column

[7]:
```
for col in tree.columns:
    print("Column", col, "has", tree[col].nunique(), "unique values")
```

```
Column Elevation has 1978 unique values
Column Aspect has 361 unique values
```

```
Column Slope has 67 unique values
Column Horizontal_Distance_To_Hydrology has 551 unique values
Column Vertical_Distance_To_Hydrology has 700 unique values
Column Horizontal_Distance_To_Roadways has 5785 unique values
Column Hillshade_9am has 207 unique values
Column Hillshade_Noon has 185 unique values
Column Hillshade_3pm has 255 unique values
Column Horizontal_Distance_To_Fire_Points has 5827 unique values
Column Wilderness_Area1 has 2 unique values
Column Wilderness_Area2 has 2 unique values
Column Wilderness_Area3 has 2 unique values
Column Wilderness_Area4 has 2 unique values
Column Soil_Type1 has 2 unique values
Column Soil_Type2 has 2 unique values
Column Soil_Type3 has 2 unique values
Column Soil_Type4 has 2 unique values
Column Soil_Type5 has 2 unique values
Column Soil_Type6 has 2 unique values
Column Soil_Type7 has 2 unique values
Column Soil_Type8 has 2 unique values
Column Soil_Type9 has 2 unique values
Column Soil_Type10 has 2 unique values
Column Soil_Type11 has 2 unique values
Column Soil_Type12 has 2 unique values
Column Soil_Type13 has 2 unique values
Column Soil_Type14 has 2 unique values
Column Soil_Type15 has 2 unique values
Column Soil_Type16 has 2 unique values
Column Soil_Type17 has 2 unique values
Column Soil_Type18 has 2 unique values
Column Soil_Type19 has 2 unique values
Column Soil_Type20 has 2 unique values
Column Soil_Type21 has 2 unique values
Column Soil_Type22 has 2 unique values
Column Soil_Type23 has 2 unique values
Column Soil_Type24 has 2 unique values
Column Soil_Type25 has 2 unique values
Column Soil_Type26 has 2 unique values
Column Soil_Type27 has 2 unique values
Column Soil_Type28 has 2 unique values
Column Soil_Type29 has 2 unique values
Column Soil_Type30 has 2 unique values
Column Soil_Type31 has 2 unique values
Column Soil_Type32 has 2 unique values
Column Soil_Type33 has 2 unique values
Column Soil_Type34 has 2 unique values
Column Soil_Type35 has 2 unique values
Column Soil_Type36 has 2 unique values
```

```
Column Soil_Type37 has 2 unique values
Column Soil_Type38 has 2 unique values
Column Soil_Type39 has 2 unique values
Column Soil_Type40 has 2 unique values
Column Cover_Type has 7 unique values
```

1.0.10 Summary results :

- `"Elevation"`, `"Slope"`, `"Horizontal_Distance_To_Hydrology"`, `"Vertical_Distance_To_Hydrology"`, `"Horizontal_Distance_To_Roadways"`, `"Horizontal_Distance_To_Fire_Points"` are continuous variables and their values vary.
- `"Aspect"` is also continuous and its values vary from 0 to 360. It has angular values.
- `"Hillshade_3pm"`, `"Hillshade_Noon"`, `"Hillshade_3pm"` are also continuos and their values vary from 0 to 255. This means that the values represent bitwise value. I concluded that the values are RGB color representation of the shadow at a particular time.
- `Wilderness_Areas` and `Soil_Types` are categorical (binary 1 or 0) data.

1.0.11 I will focus on columns in terms of their values (categorical or continuous)

1.0.12 "Cover_Type"

[8]:
```python
# There are 7 types of trees in the forest district.

tree.Cover_Type.value_counts()
```

[8]:
```
2    283301
1    211840
3     35754
7     20510
6     17367
5      9493
4      2747
Name: Cover_Type, dtype: int64
```

1.0.13 Handling with Outliers

- The columns which have continuous value should be examined in terms of outliers.
- First, I will choose the columns which have unique values more than 7.
- Second, I will define two functions to help me understand the outliers and how I can handle with them.
- Third, I will define another function to detect outliers in accordance with the `zscore` (how many times IQR) value I choose according to the result from the previous functions.

- Lastly, I will drop rows which have outliers.

Selecting Continuous Columns which have More than 7 (Why 7? Based on the categorical data number of `Cover_Type`) Unique Values

[9]:
```
numeric = []

for col in tree.columns:
    if tree[col].nunique() > 7 : numeric.append(col)
print(numeric)
```

```
['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways',
'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm',
'Horizontal_Distance_To_Fire_Points']
```

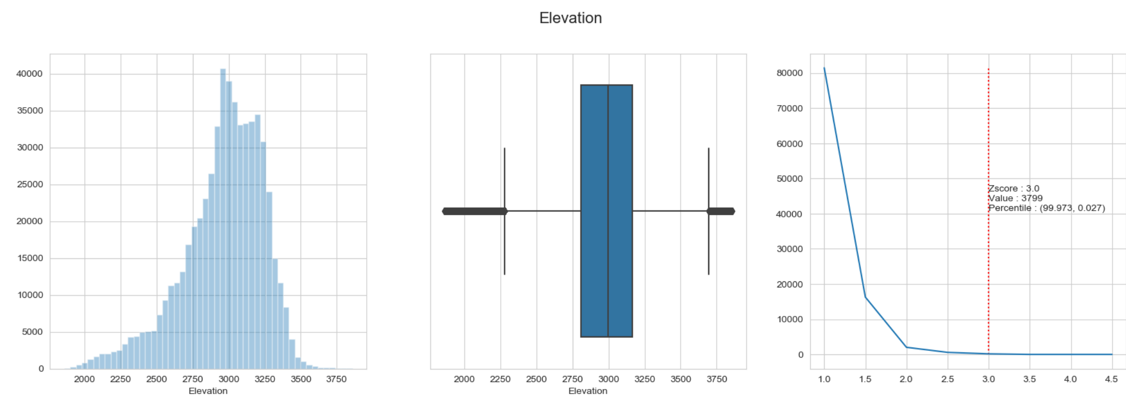
Defining functions to determine zscore in order to specify IQR Distance

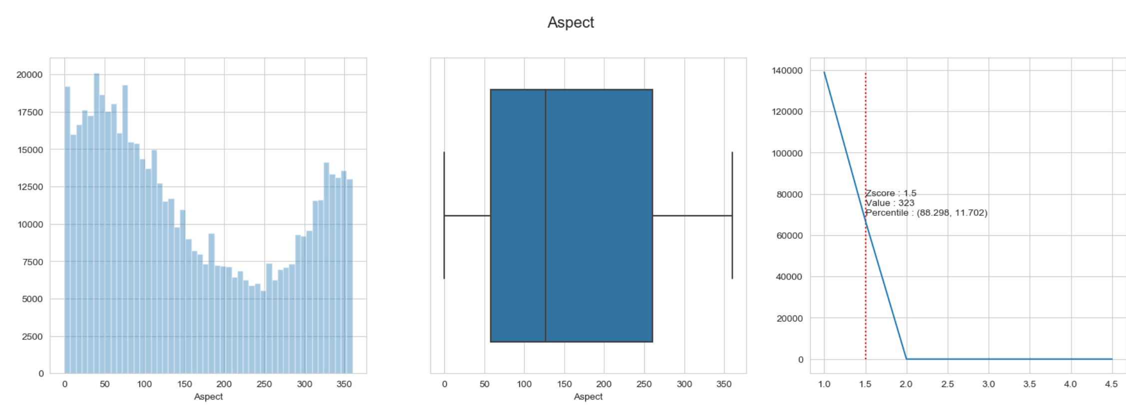
[10]:
```
def outlier_zscore(df, col, min_z=1, max_z = 5, step = 0.1, print_list = False):
    z_scores = zscore(df[col].dropna())
    threshold_list = []
    for threshold in np.arange(min_z, max_z, step):
        threshold_list.append((threshold, len(np.where(z_scores >␣
↪threshold)[0])))
        df_outlier = pd.DataFrame(threshold_list, columns = ['threshold',␣
↪'outlier_count'])
        df_outlier['pct'] = (df_outlier.outlier_count - df_outlier.
↪outlier_count.shift(-1))/df_outlier.outlier_count*100
    plt.plot(df_outlier.threshold, df_outlier.outlier_count)
    best_treshold = round(df_outlier.iloc[df_outlier.pct.argmax(), 0],2)
    outlier_limit = int(df[col].dropna().mean() + (df[col].dropna().std()) *␣
↪df_outlier.iloc[df_outlier.pct.argmax(), 0])
    percentile_threshold = stats.percentileofscore(df[col].dropna(),␣
↪outlier_limit)
    plt.vlines(best_treshold, 0, df_outlier.outlier_count.max(),
               colors="r", ls = ":"
              )
    plt.annotate("Zscore : {}\nValue : {}\nPercentile : {}".
↪format(best_treshold, outlier_limit,
                                                                          (np.
↪round(percentile_threshold, 3),
                                                                           np.
↪round(100-percentile_threshold, 3))),
                 (best_treshold, df_outlier.outlier_count.max()/2))
    #plt.show()
    if print_list:
        print(df_outlier)
    return (plt, df_outlier, best_treshold, outlier_limit, percentile)
```

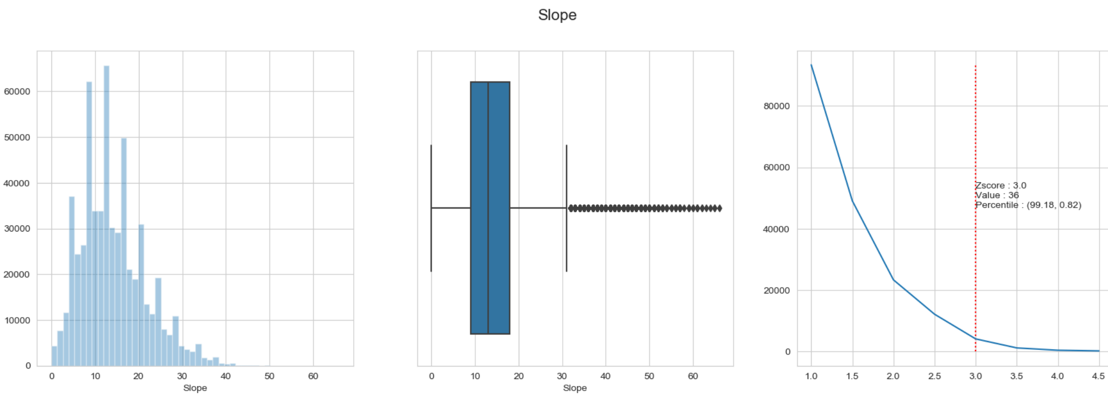
```
[11]: def outlier_inspect(df, col, min_z=1, max_z = 5, step = 0.5, max_hist = None,␣
       ↪bins = 50):
          fig = plt.figure(figsize=(20, 6))
          fig.suptitle(col, fontsize=16)
          plt.subplot(1,3,1)
          if max_hist == None:
              sns.distplot(df[col], kde=False, bins = 50)
          else :
              sns.distplot(df[df[col]<=max_hist][col], kde=False, bins = 50)
          plt.subplot(1,3,2)
          sns.boxplot(df[col])
          plt.subplot(1,3,3)
          z_score_inspect = outlier_zscore(df, col, min_z=min_z, max_z = max_z, step␣
       ↪= step)
          plt.show()
```

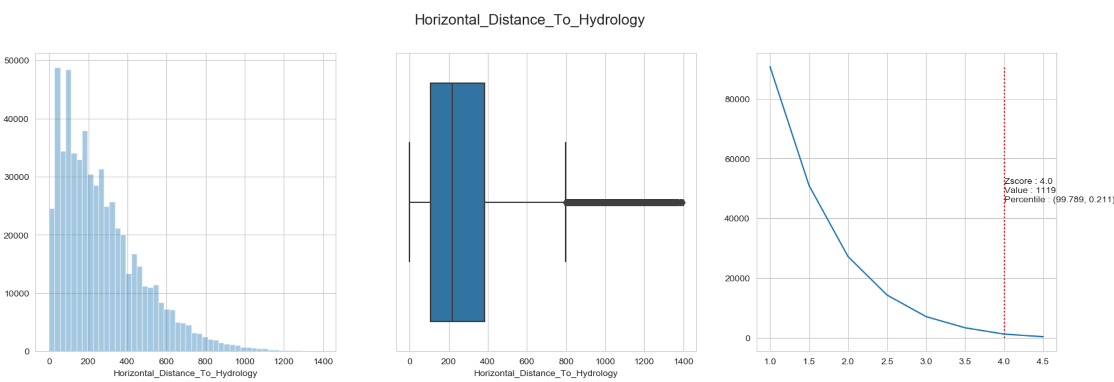
Visualization of Columns' Outliers

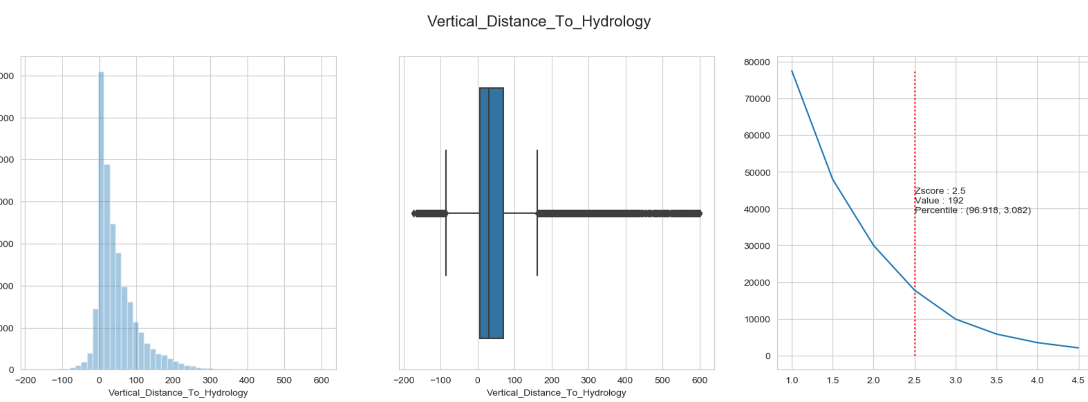
```
[12]: for col in numeric:
          outlier_inspect(tree, col)
```



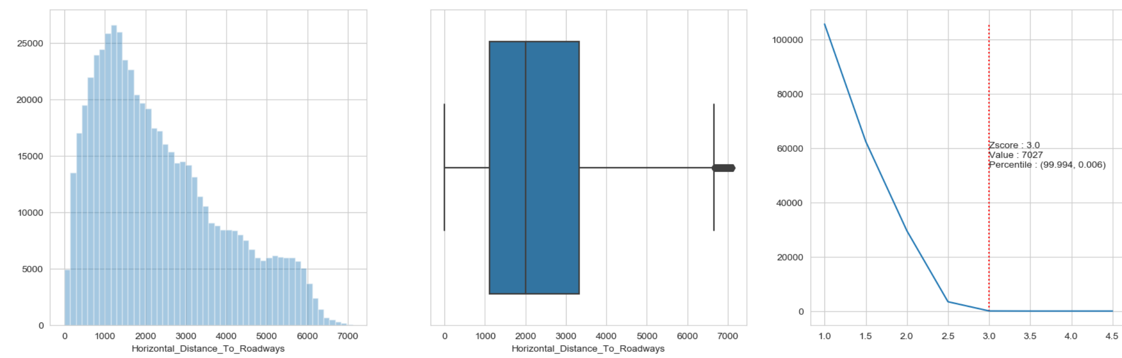



Slope
Slope
Slope
Zscore : 3.0
Value : 36
Percentile : (99.18, 0.82)

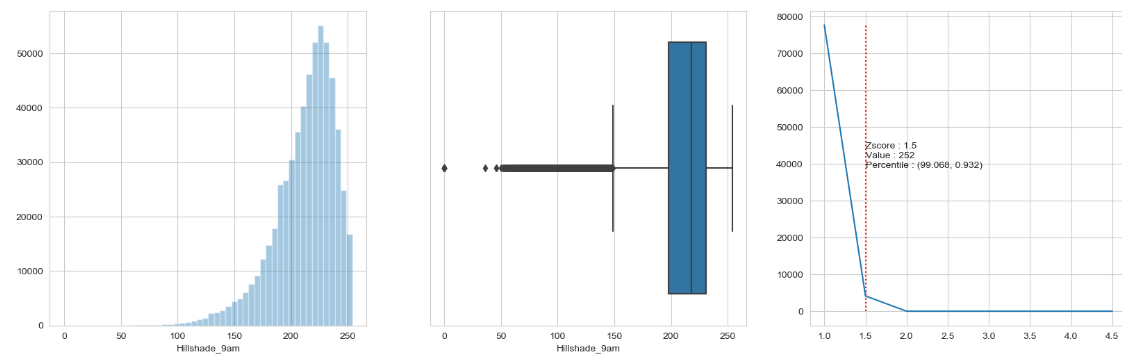

Horizontal_Distance_To_Hydrology
Horizontal_Distance_To_Hydrology
Horizontal_Distance_To_Hydrology
Zscore : 4.0
Value : 1119
Percentile : (99.789, 0.211)


Vertical_Distance_To_Hydrology
Vertical_Distance_To_Hydrology
Vertical_Distance_To_Hydrology
Zscore : 2.5
Value : 192
Percentile : (96.918, 3.082)

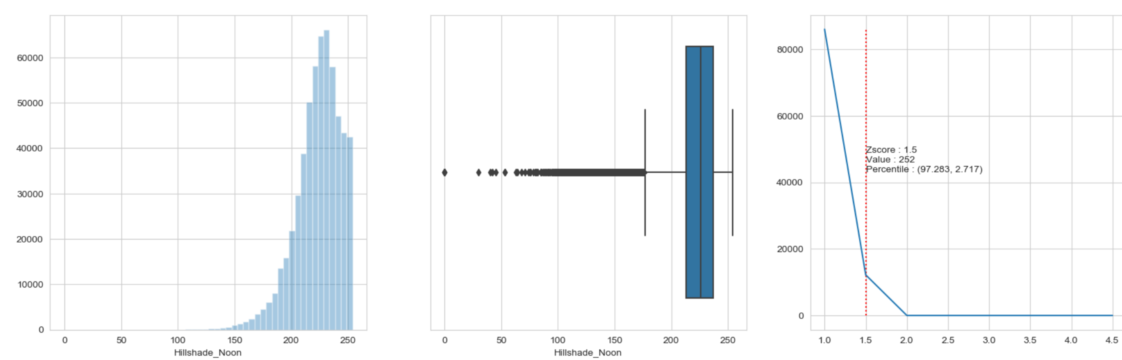
Horizontal_Distance_To_Roadways

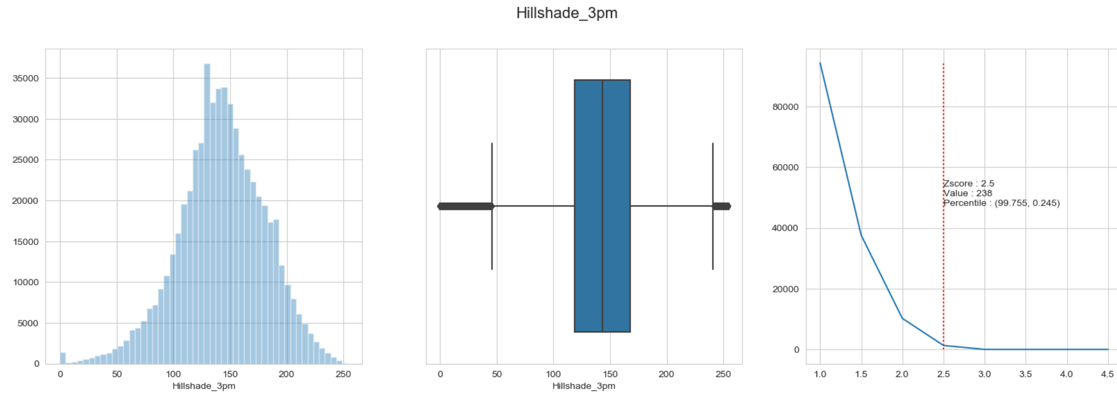


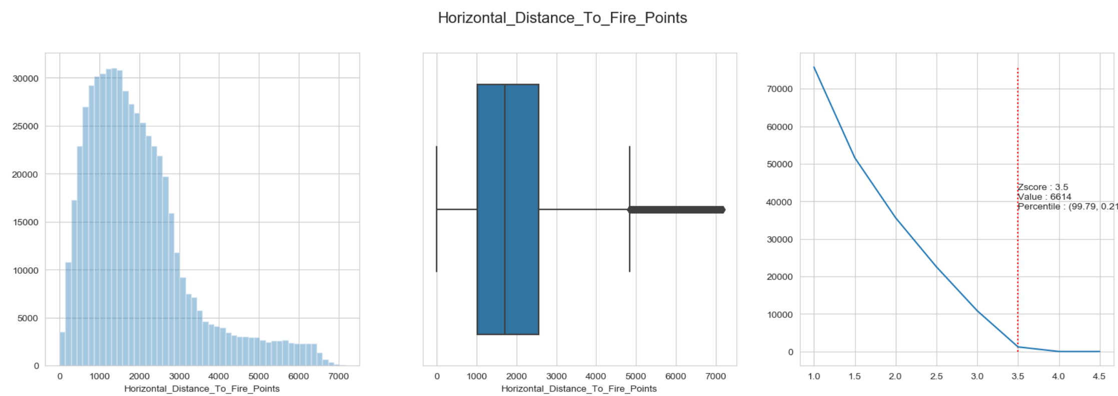
Hillshade_9am



Hillshade_Noon





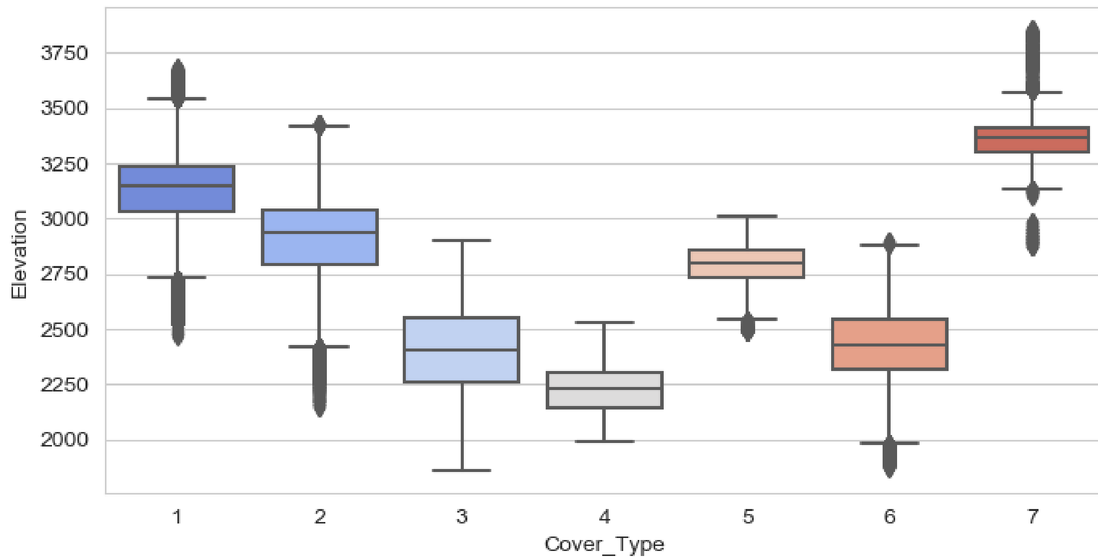


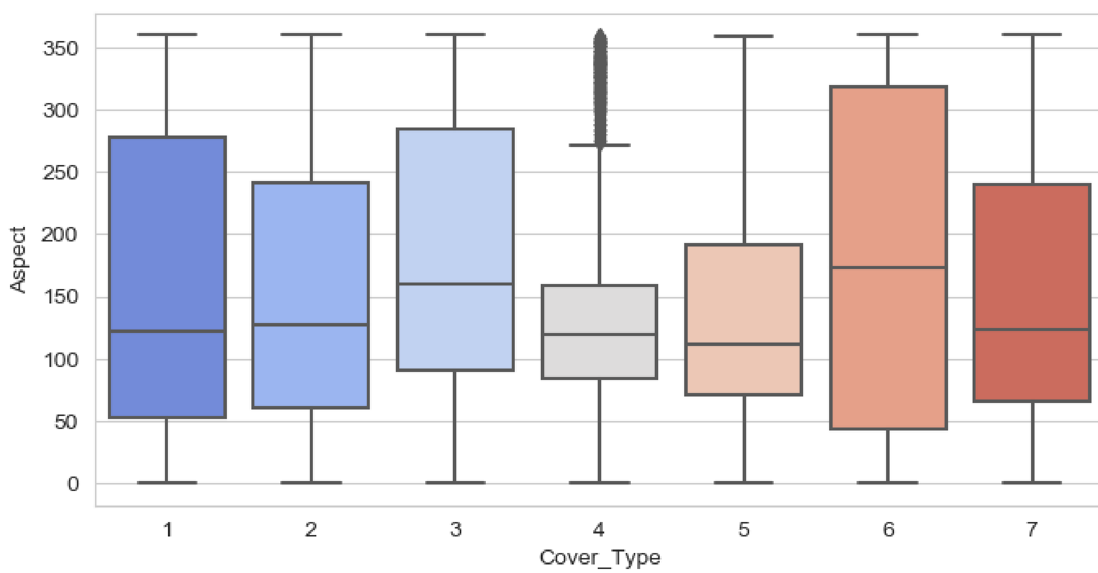
1.0.14 Summary results :

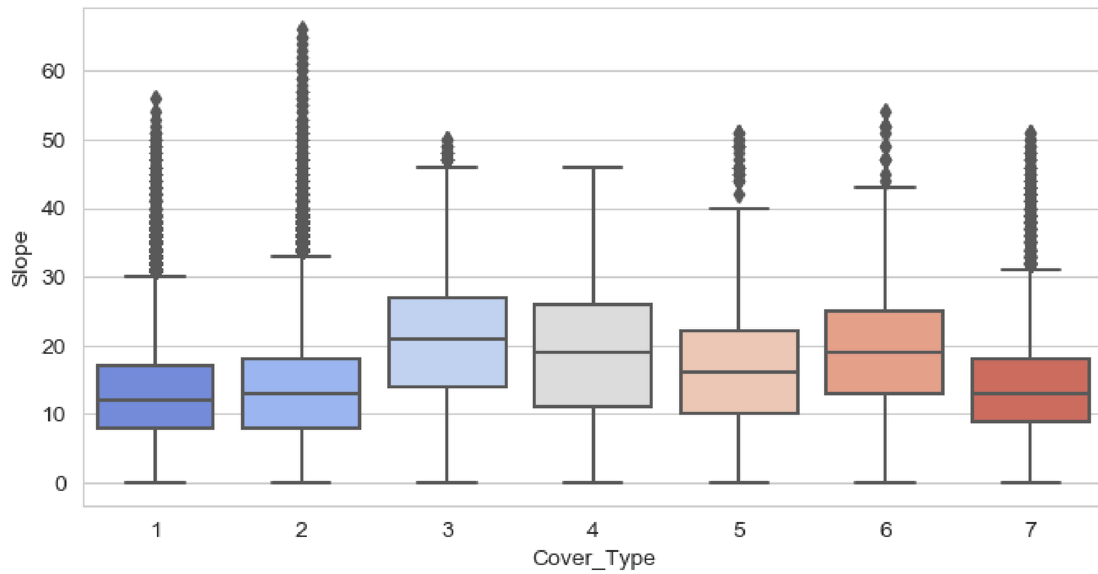
- If you focus on the plots on right side, you should realize that vertical red line indicates the `IQR` distance (`zscore`).
- The `zscore` values vary from `1.5` to `4`. So I will choose `3` as `IQR` distance.
- If your focus on the plots on the left side, you can see the skewness of each continuous column.

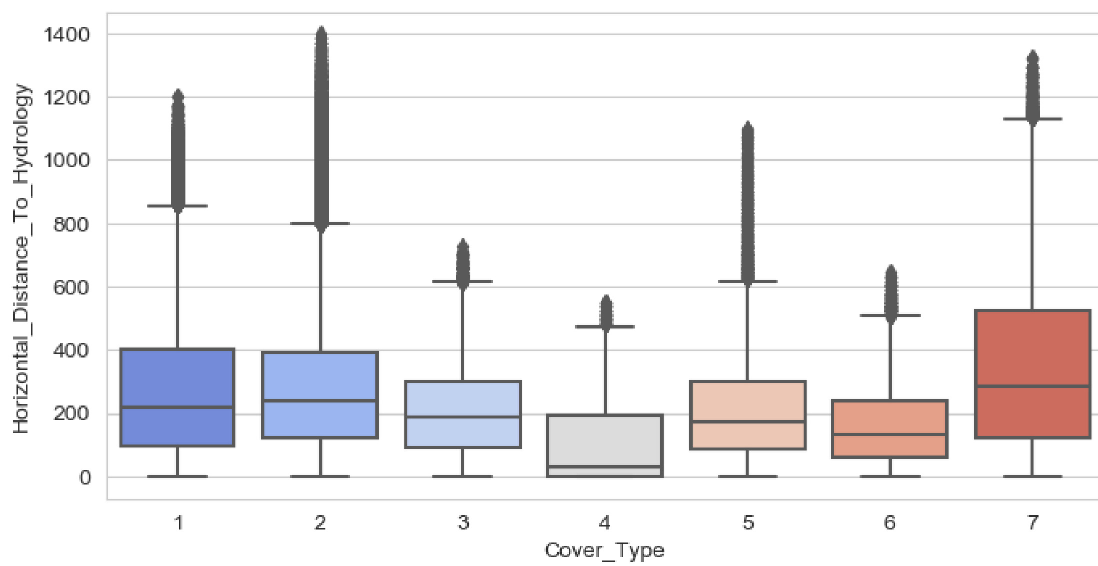
Now, I want to check the outliers shape of continous features with respect to the target (`Cover_Type`) classes.

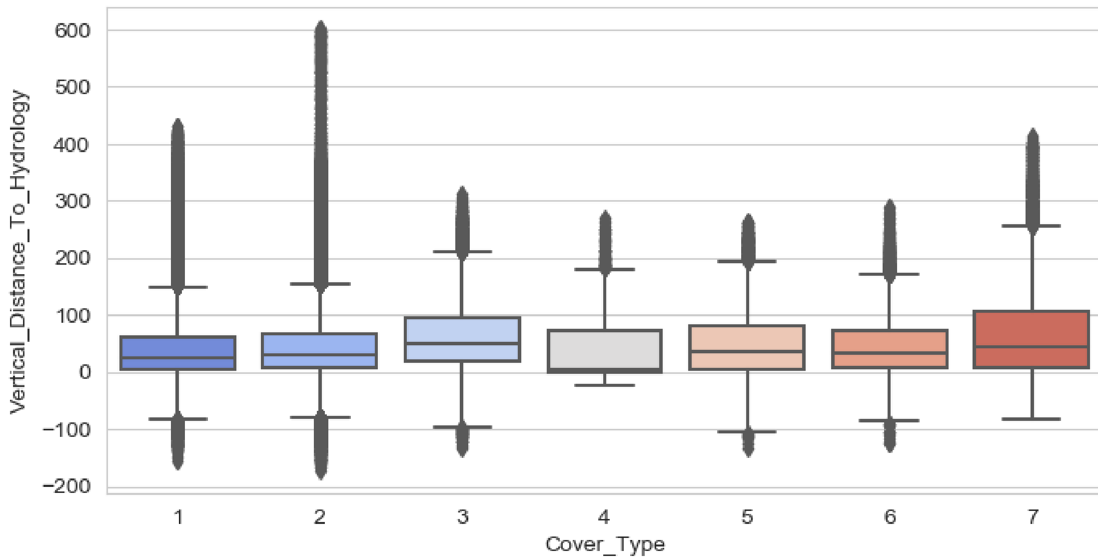
[13]:
```
tree['Cover_Type'] = tree['Cover_Type'].astype('category') #To convert target␣
↪class into category

for i, col in enumerate(numeric):
    plt.figure(i, figsize=(8,4))
    sns.boxplot(x = tree['Cover_Type'], y=col, data=tree, palette="coolwarm")
```

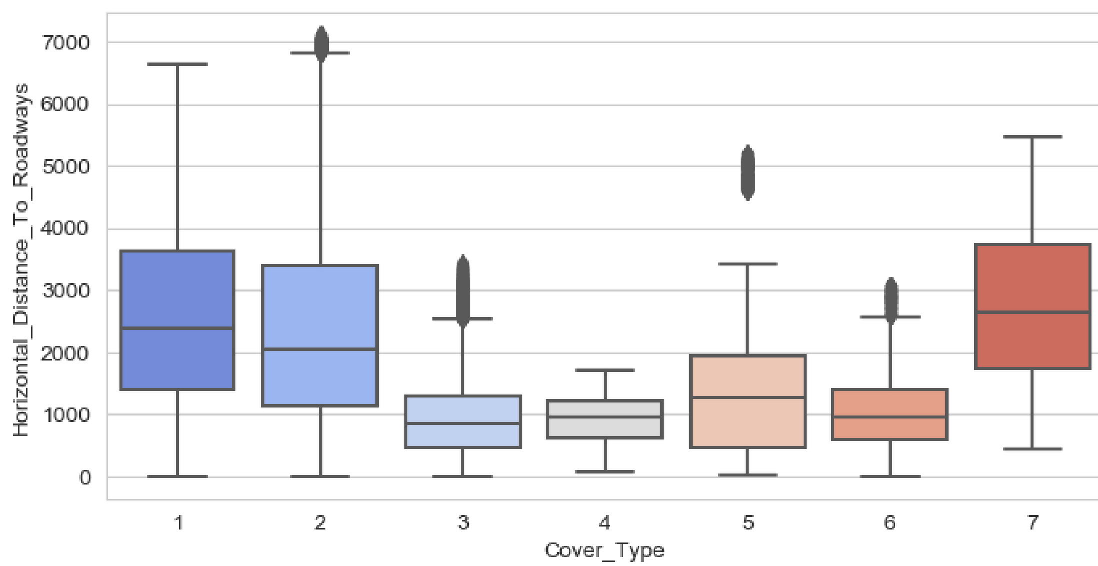

3750
3500
3250
3000
2750
2500
2250
2000
Elevation
1
2
3
4
5
6
7
Cover_Type

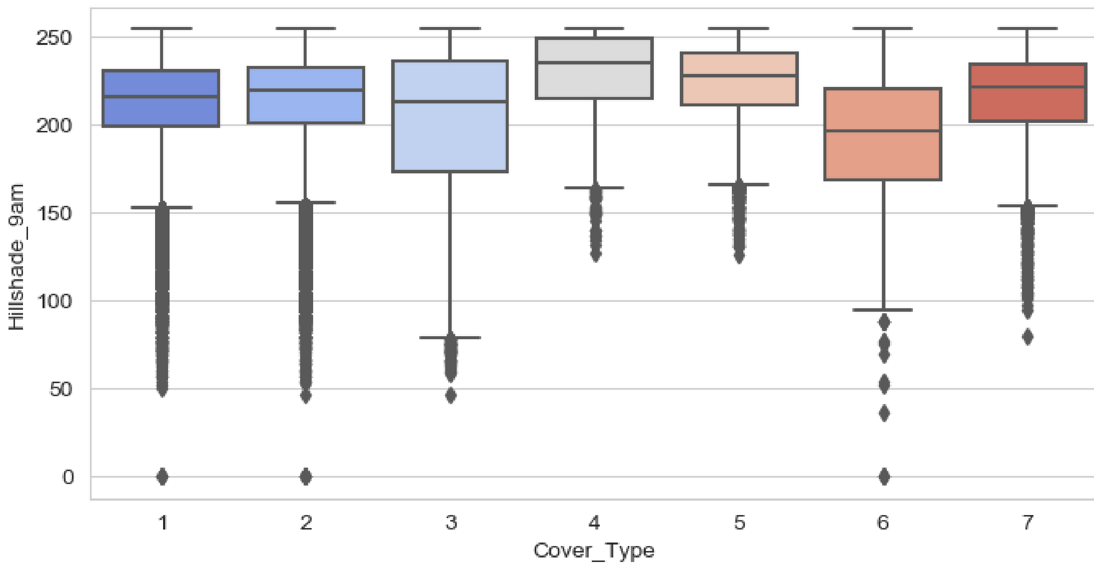

350
300
250
200
150
100
50
0
Aspect
1
2
3
4
5
6
7
Cover_Type

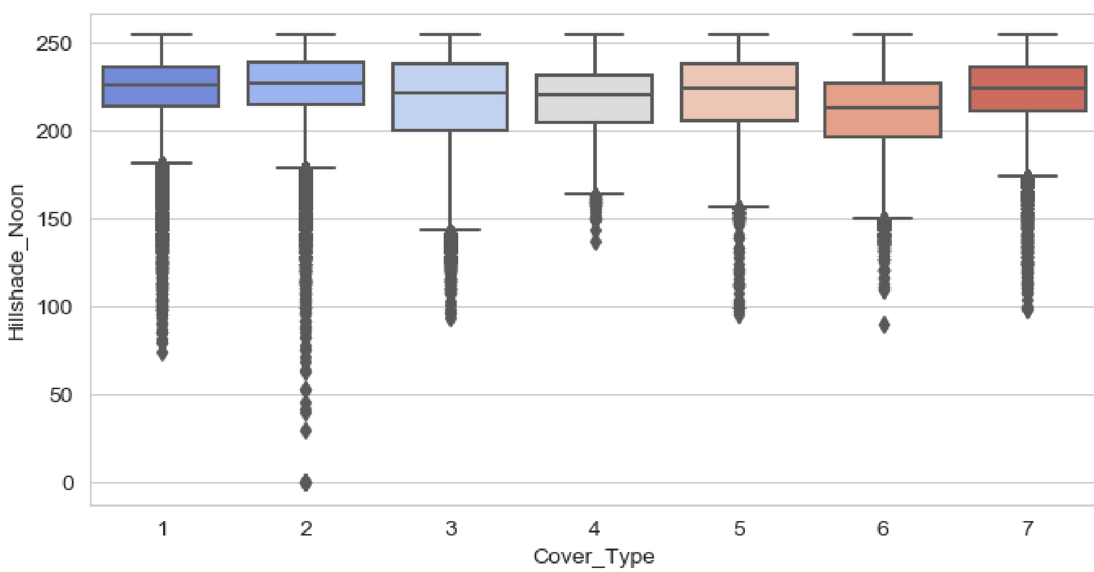

Slope
0
10
20
30
40
50
60
1
2
3
4
5
6
7
Cover_Type

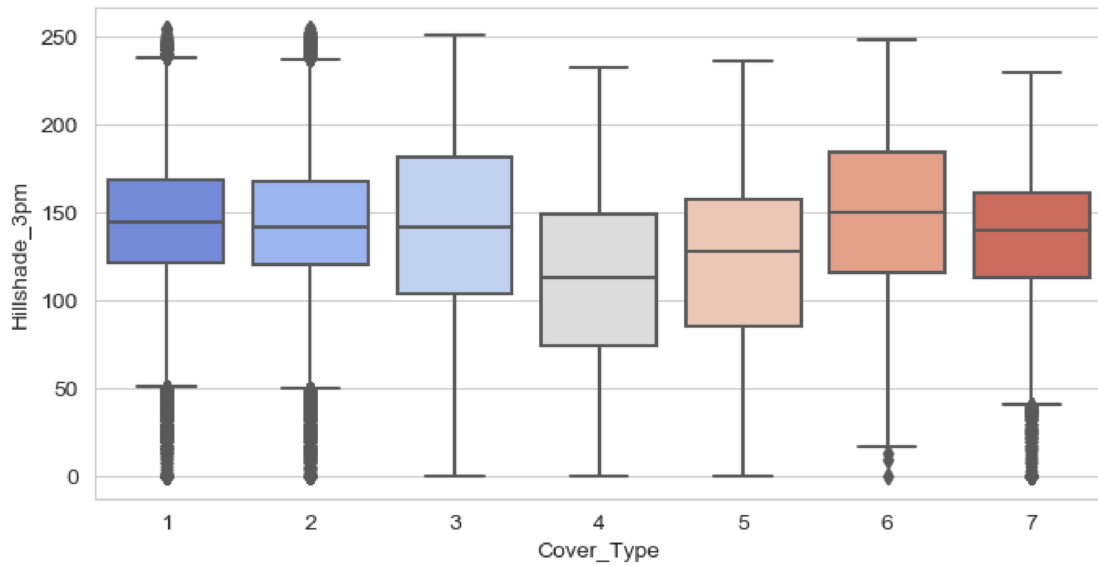

Horizontal_Distance_To_Hydrology
0
200
400
600
800
1000
1200
1400
1
2
3
4
5
6
7
Cover_Type

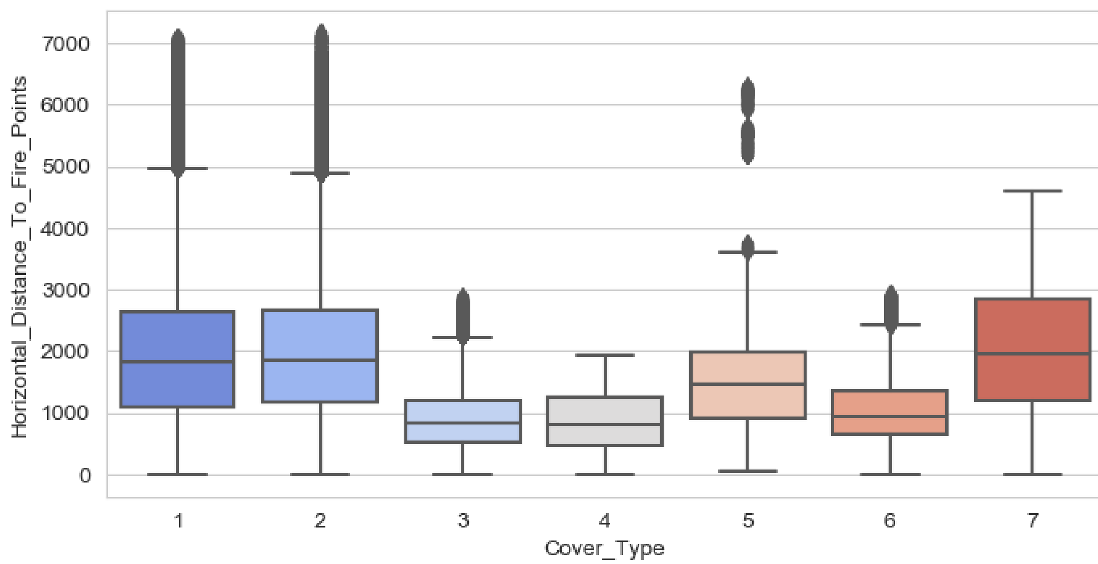

600
500
400
300
200
100
0
−100
−200
Vertical_Distance_To_Hydrology
1
2
3
4
5
6
7
Cover_Type


7000
6000
5000
4000
3000
2000
1000
0
Horizontal_Distance_To_Roadways
1
2
3
4
5
6
7
Cover_Type


250
200
150
100
50
0
Hillshade_9am
1
2
3
4
5
6
7
Cover_Type


250
200
150
100
50
0
Hillshade_Noon
1
2
3
4
5
6
7
Cover_Type





[14]:
```
def detect_outliers(df, col_name):
    '''
    this function detects outliers based on 3 time IQR and
    returns the number of lower and uper limit and number of outliers␣
↪respectively
    '''
    first_quartile = np.percentile(np.array(df[col_name].tolist()), 25)
    third_quartile = np.percentile(np.array(df[col_name].tolist()), 75)
```

```
    IQR = third_quartile - first_quartile

    upper_limit = third_quartile+(3*IQR)
    lower_limit = first_quartile-(3*IQR)
    outlier_count = 0

    for value in df[col_name].tolist():
        if (value < lower_limit) | (value > upper_limit):
            outlier_count +=1
    return lower_limit, upper_limit, outlier_count
```

Let's see how many outliers are there of each Continuous Columns

[15]:
```
for col in numeric:
    if detect_outliers(tree, col)[2] > 0:
        print("There are {} outliers in {}".format(detect_outliers(tree,␣
 ↪col)[2], col))
```

```
There are 275 outliers in Slope
There are 414 outliers in Horizontal_Distance_To_Hydrology
There are 5339 outliers in Vertical_Distance_To_Hydrology
There are 1027 outliers in Hillshade_9am
There are 1191 outliers in Hillshade_Noon
There are 10 outliers in Horizontal_Distance_To_Fire_Points
```

1.0.15 According to the plotting and analysis above, we should focus on the following columns in terms of outliers :

- There are 275 outliers in `Slope`
- There are 414 outliers in `Horizontal_Distance_To_Hydrology`
- There are 5339 outliers in `Vertical_Distance_To_Hydrology`
- There are 10 outliers in `Horizontal_Distance_To_Fire_Points`

1.0.16 I will drop all rows which contain outliers in these four colums above

[16]:
```
tree1 = tree[(tree['Slope'] > detect_outliers(tree, 'Slope')[0]) &
              (tree['Slope'] < detect_outliers(tree, 'Slope')[1])]
tree1.shape
```

[16]:
```
(580640, 55)
```

[17]:
```
tree1 = tree1[(tree1['Horizontal_Distance_To_Fire_Points'] >␣
 ↪detect_outliers(tree1, 'Horizontal_Distance_To_Fire_Points')[0]) &
              (tree1['Horizontal_Distance_To_Fire_Points'] <␣
 ↪detect_outliers(tree1, 'Horizontal_Distance_To_Fire_Points')[1])]
```

```
tree1.shape
```

[17]: (580630, 55)

[18]:
```
tree1 = tree1[(tree1['Horizontal_Distance_To_Hydrology'] >␣
↪detect_outliers(tree1, 'Horizontal_Distance_To_Hydrology')[0]) &
              (tree1['Horizontal_Distance_To_Hydrology'] <␣
↪detect_outliers(tree1, 'Horizontal_Distance_To_Hydrology')[1])]

tree1.shape
```

[18]: (580216, 55)

[19]:
```
tree1 = tree1[(tree1['Vertical_Distance_To_Hydrology'] > detect_outliers(tree1,␣
↪'Vertical_Distance_To_Hydrology')[0]) &
              (tree1['Vertical_Distance_To_Hydrology'] < detect_outliers(tree1,␣
↪'Vertical_Distance_To_Hydrology')[1])]

tree1.shape
```

[19]: (574967, 55)

We dropped 6045 rows totally which contain outlier values

[20]:
```
len(tree) - len(tree1)
```

[20]: 6045

[21]:
```
tree1 = tree1.reset_index(drop=True)
```

1.0.17 Continue to EDA

- I should check the relation between `Wilderness Areas` & `Soil Types` in terms of Types of Trees(`Cover_Type`)

First, `Wilderness_Areas` :

[22]:
```
# it belongs to only one "Wilderness_Area"s at the same time.

total_area = tree1["Wilderness_Area1"] + tree1["Wilderness_Area2"] +␣
↪tree1["Wilderness_Area3"] + tree1["Wilderness_Area4"]
print(total_area.value_counts())
```

```
1    574967
dtype: int64
```

Second, `Soil_Types` :

[23]:
```
soil_columns = [i for i in tree1.columns if "Soil" in i]
```

[24]:
```
# sum of all soil type columns' values
total_soil = 0
for x in [i for i in tree1.columns if "Soil" in i]:
    total_soil += tree1[x]
```

[25]:
```
# it belongs to only one "Soil_Type"s at the same time.

print(total_soil.value_counts())
```

```
1    574967
Name: Soil_Type1, dtype: int64
```

1.0.18 Summary results :

- Yes, the analysis above prove that `Wilderness_Areas` and `Soil_Types` columns have only `1` or `0` and it belongs to only one `Soil_Type` or `Wilderness_Area` at the same time.
- These columns are `get_dummied` (one-hot encoded) from categorical values.

1.0.19 I want to explore the number of values counts within each Binary types

[26]:
```
binaries = tree1.loc[:,'Wilderness_Area1':'Soil_Type40']
```

[27]:
```
for col in binaries:
    count = binaries[col].value_counts()[1]
    print(col, count)
```

```
Wilderness_Area1 259664
Wilderness_Area2 29865
Wilderness_Area3 248652
Wilderness_Area4 36786
Soil_Type1 3017
Soil_Type2 7511
Soil_Type3 4818
Soil_Type4 12314
Soil_Type5 1597
Soil_Type6 6556
Soil_Type7 105
Soil_Type8 179
Soil_Type9 1147
Soil_Type10 32418
```

```
Soil_Type11 12382
Soil_Type12 29957
Soil_Type13 16871
Soil_Type14 599
Soil_Type15 3
Soil_Type16 2845
Soil_Type17 3422
Soil_Type18 1899
Soil_Type19 4021
Soil_Type20 9259
Soil_Type21 838
Soil_Type22 33370
Soil_Type23 57698
Soil_Type24 21063
Soil_Type25 474
Soil_Type26 2589
Soil_Type27 792
Soil_Type28 766
Soil_Type29 114948
Soil_Type30 29960
Soil_Type31 25236
Soil_Type32 51895
Soil_Type33 43783
Soil_Type34 1583
Soil_Type35 1891
Soil_Type36 119
Soil_Type37 298
Soil_Type38 15533
Soil_Type39 13668
Soil_Type40 7543
```

I decided to choose a limit of just over 1000 values. Let's examine negligible `Soil_Types` that have less than 1100 values.

[28]:
```
[[col, binaries[col].value_counts()[1]] for col in binaries if [col,␣
↪binaries[col].value_counts()[1]][1]<1100]
```

[28]:
```
[['Soil_Type7', 105],
 ['Soil_Type8', 179],
 ['Soil_Type14', 599],
 ['Soil_Type15', 3],
 ['Soil_Type21', 838],
 ['Soil_Type25', 474],
 ['Soil_Type27', 792],
 ['Soil_Type28', 766],
 ['Soil_Type36', 119],
 ['Soil_Type37', 298]]
```

[29]:
```python
[col for col in binaries if [col, binaries[col].value_counts()[1]][1]<1000]
```

[29]:
```
['Soil_Type7',
 'Soil_Type8',
 'Soil_Type14',
 'Soil_Type15',
 'Soil_Type21',
 'Soil_Type25',
 'Soil_Type27',
 'Soil_Type28',
 'Soil_Type36',
 'Soil_Type37']
```

- There are some of the `Soil types` which consists of very few counts above. I will drop these columns before modeling in Feature Engineering section using SQL.

1.0.20 My target column is `Cover_Type`. So let's take a close look at this column.

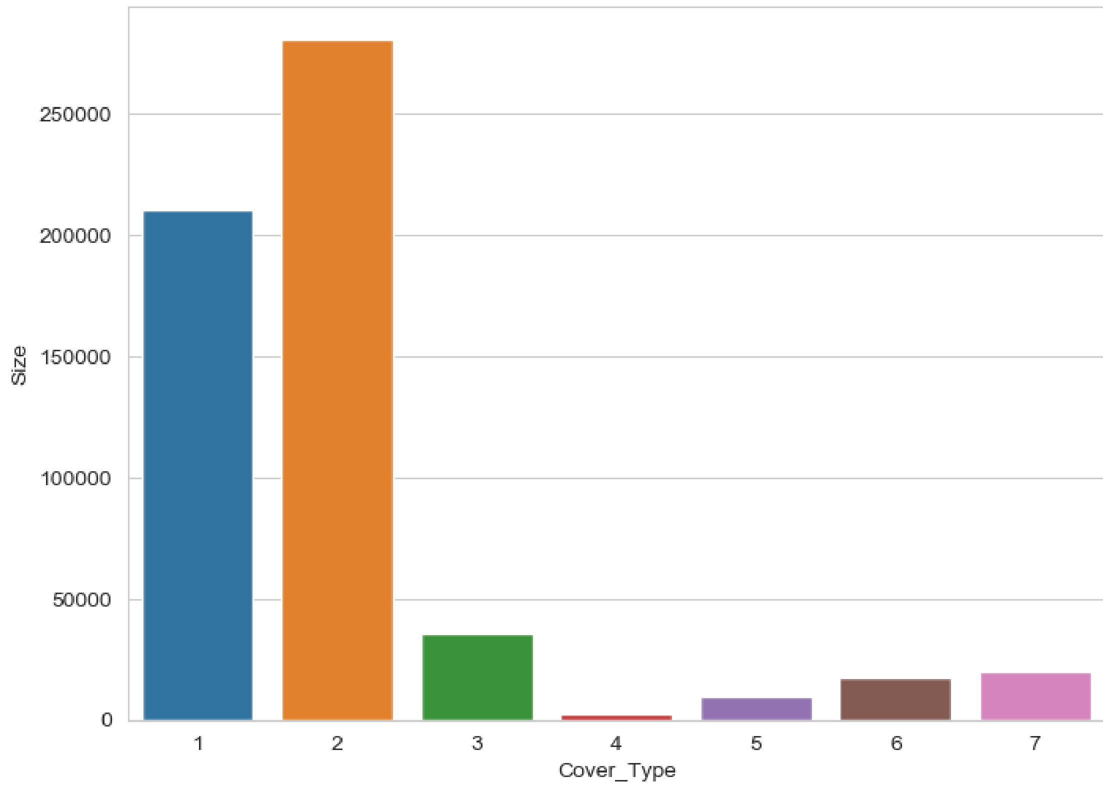
The `Cover_Type` column has unbalanced labels of values. The value distance is far away between type of "1" & "2" and the others.

[30]:
```python
for i in range(1,8) :
    print("the shape of the value of", i, tree1[tree1["Cover_Type"] == i].shape)
```

```
the shape of the value of 1 (210004, 55)
the shape of the value of 2 (280193, 55)
the shape of the value of 3 (35546, 55)
the shape of the value of 4 (2741, 55)
the shape of the value of 5 (9453, 55)
the shape of the value of 6 (17345, 55)
the shape of the value of 7 (19685, 55)
```

[31]:
```python
class_tree = tree1.groupby('Cover_Type').size()
class_label = pd.DataFrame(class_tree,columns = ['Size'])
plt.figure(figsize = (8,6))
sns.barplot(x = class_label.index, y = 'Size', data = class_label)
```

[31]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x283a24f9588>
```



- We can see that we have unbalanced data (`Cover_Type`). But, additionally I would like to check the distribution of each class of `Cover_Type` in terms of percentages.

[32]:
```
for i, number in enumerate(class_tree):
    percent = (number/class_tree.sum())
    print('Cover_Type', class_tree.index[i])
    print('%.2f'% percent)
```

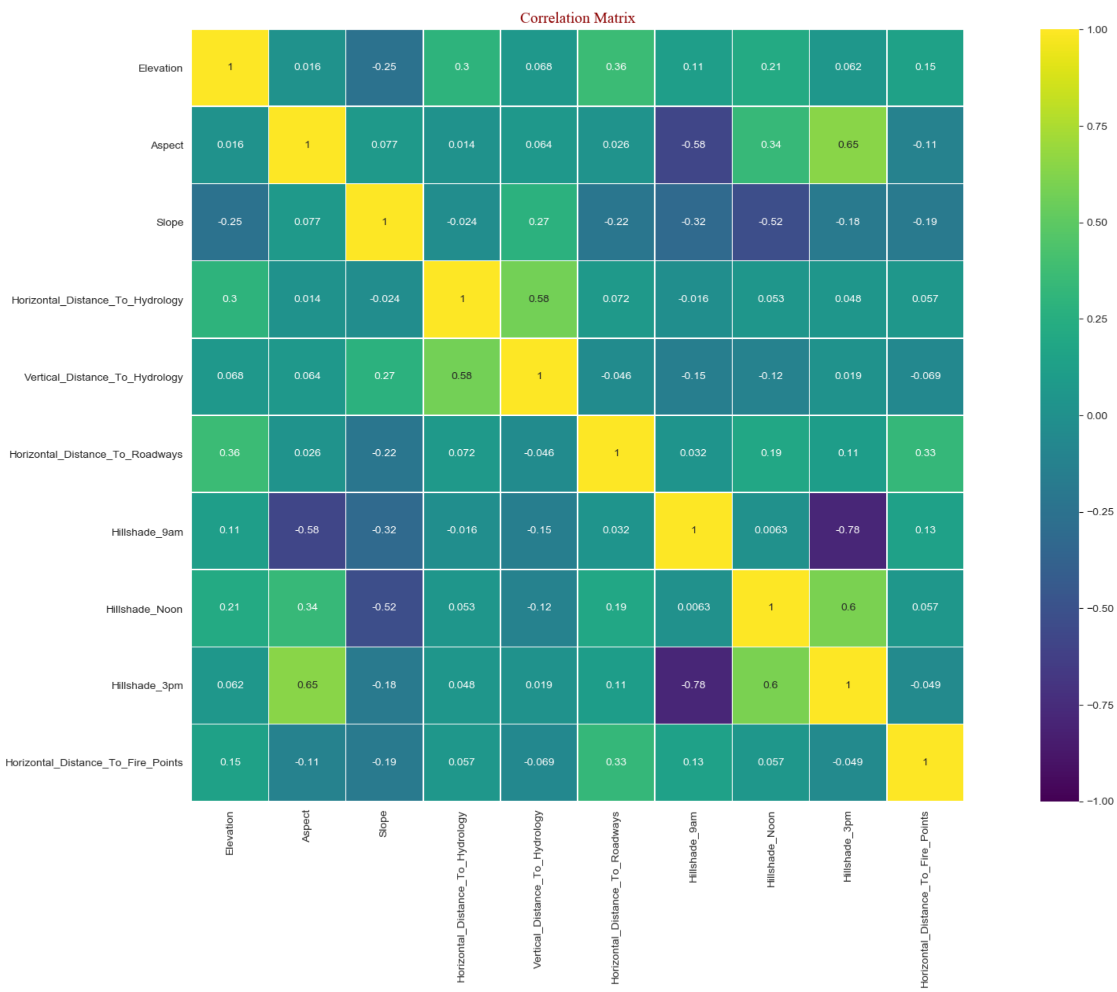
```
Cover_Type 1
0.37
Cover_Type 2
0.49
Cover_Type 3
0.06
Cover_Type 4
0.00
Cover_Type 5
0.02
Cover_Type 6
0.03
Cover_Type 7
0.03
```

1.0.21 Summary result:

- I decided to keep the `Cover_Type` column as it is to see the different results in terms of the various ML Models' behavior to unbalanced data.

1.0.22 Now, let's take a closer look at correlation of continuous columns.

[33]:
```
corr_matrix = tree1[numeric].corr()

plt.figure(figsize=(25, 13))
sns.heatmap(corr_matrix,square=True, annot=True, linewidths=.5, vmin=-1,␣
↪vmax=1, cmap='viridis')
plt.title("Correlation Matrix", fontdict=font_title)

plt.show()
```



1.0.23 Summary results :

- `Hillshade_3pm` and `Hillshade_9am` are highly correlated. So I decided to drop `Hillshade_3pm`.
- `Horizontal_Distance_To_Hydrology` and `Vertical_Distance_To_Hydrology` columns somehow are not correlated (0.58) enough, so I decided to transform a new column derived from these two columns.
- `Horizontal_Distance_To_Hydrology` and `Horizontal_Distance_To_Roadways` are not correlated (0.072) , so I decided to transform a new column derived from these two columns.
- `Vertical_Distance_To_Hydrology` and `Elevation` are not correlated (0.068) , so I decided to transform a new column derived from these two columns.

So far, we have implemented EDA process to recognize and analyze our dataset and made some decisions to adapt it to our models.

[34]:
```
tree1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 574967 entries, 0 to 574966
Data columns (total 55 columns):
 #   Column                              Non-Null Count   Dtype
---  ------                              --------------   -----
 0   Elevation                           574967 non-null  int64
 1   Aspect                              574967 non-null  int64
 2   Slope                               574967 non-null  int64
 3   Horizontal_Distance_To_Hydrology    574967 non-null  int64
 4   Vertical_Distance_To_Hydrology      574967 non-null  int64
 5   Horizontal_Distance_To_Roadways     574967 non-null  int64
 6   Hillshade_9am                       574967 non-null  int64
 7   Hillshade_Noon                      574967 non-null  int64
 8   Hillshade_3pm                       574967 non-null  int64
 9   Horizontal_Distance_To_Fire_Points  574967 non-null  int64
 10  Wilderness_Area1                    574967 non-null  int64
 11  Wilderness_Area2                    574967 non-null  int64
 12  Wilderness_Area3                    574967 non-null  int64
 13  Wilderness_Area4                    574967 non-null  int64
 14  Soil_Type1                          574967 non-null  int64
 15  Soil_Type2                          574967 non-null  int64
 16  Soil_Type3                          574967 non-null  int64
 17  Soil_Type4                          574967 non-null  int64
 18  Soil_Type5                          574967 non-null  int64
 19  Soil_Type6                          574967 non-null  int64
 20  Soil_Type7                          574967 non-null  int64
 21  Soil_Type8                          574967 non-null  int64
 22  Soil_Type9                          574967 non-null  int64
 23  Soil_Type10                         574967 non-null  int64
```

```
 24  Soil_Type11                         574967 non-null  int64
 25  Soil_Type12                         574967 non-null  int64
 26  Soil_Type13                         574967 non-null  int64
 27  Soil_Type14                         574967 non-null  int64
 28  Soil_Type15                         574967 non-null  int64
 29  Soil_Type16                         574967 non-null  int64
 30  Soil_Type17                         574967 non-null  int64
 31  Soil_Type18                         574967 non-null  int64
 32  Soil_Type19                         574967 non-null  int64
 33  Soil_Type20                         574967 non-null  int64
 34  Soil_Type21                         574967 non-null  int64
 35  Soil_Type22                         574967 non-null  int64
 36  Soil_Type23                         574967 non-null  int64
 37  Soil_Type24                         574967 non-null  int64
 38  Soil_Type25                         574967 non-null  int64
 39  Soil_Type26                         574967 non-null  int64
 40  Soil_Type27                         574967 non-null  int64
 41  Soil_Type28                         574967 non-null  int64
 42  Soil_Type29                         574967 non-null  int64
 43  Soil_Type30                         574967 non-null  int64
 44  Soil_Type31                         574967 non-null  int64
 45  Soil_Type32                         574967 non-null  int64
 46  Soil_Type33                         574967 non-null  int64
 47  Soil_Type34                         574967 non-null  int64
 48  Soil_Type35                         574967 non-null  int64
 49  Soil_Type36                         574967 non-null  int64
 50  Soil_Type37                         574967 non-null  int64
 51  Soil_Type38                         574967 non-null  int64
 52  Soil_Type39                         574967 non-null  int64
 53  Soil_Type40                         574967 non-null  int64
 54  Cover_Type                          574967 non-null  category
dtypes: category(1), int64(54)
memory usage: 237.4 MB
```

I saved the EDA-implemented dataset as `covtype_EDA` for importing into SQLite.

[41]:
```python
tree1.to_csv("covtype_EDA.csv", index = False)
```

Now, its time to implement Feature Engineering using `SQLite`

1.1 Feature Engineering

1.1.1 My Plan of Feature Extraction

- First, I decided to produce&transform a new column with `Horizontal_Distance_To_Hydrology` and `Vertical_Distance_To_Hydrology` columns. New column will contain the values of **Hypotenuse** of `horizantal` and `vertical` distances.
- As second, we can produce&transform an additional column which contains **average** of Horizantal Distances to Hydrology and Roadways.
- Third, I decided to transform a new column which contains **average** of `Elevation` and `Vertical_Distance_To_Hydrology` columns. So that, there is no need to have `Horizontal_Distance_To_Hydrology` and `Vertical_Distance_To_Hydrology` columns, because I have new columns which represent more value than them. I decide to drop these columns.
- Lastly, I will drop unnecessary columns `'Hillshade_3pm', 'Soil_Type7', 'Soil_Type8', 'Soil_Type14', 'Soil_Type15', 'Soil_Type21', 'Soil_Type25', 'Soil_Type28', 'Soil_Type36', 'Soil_Type37'`) that I conclued previously.
- Note that, after seeing the result of the models, there may be a possibility of making minor changes to the features in the modeling phase.

1.1.2 Now, let's do these process above using SQLite.

- Importing `covtype_EDA.csv` file into my local database in SQLite.
- Creating table named `"covtype_EDA"`
- Transforming new features on SQLite using the syntax below :

```sql
SELECT *,
(Horizontal_Distance_To_Hydrology*Horizontal_Distance_To_Hydrology)+(Vertical_Distance_To_Hydro
(Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways)/2 as Average_Dist_Road_Hyd
FROM covtype_EDA;
```

- Saving the table (dataset of `covtype_EDA.csv`) as new table named `covtype1` on SQLite using the syntax below :

```sql
CREATE TABLE covtype1 as SELECT *,
(Horizontal_Distance_To_Hydrology*Horizontal_Distance_To_Hydrology)+(Vertical_Distance_To_Hydro
(Horizontal_Distance_To_Hydrology + Horizontal_Distance_To_Roadways)/2 as Average_Dist_Road_Hyd
FROM covtype_EDA;
```

- Selecting following required columns in SQLite :

```sql
SELECT Elevation, Aspect, Slope, Horizontal_Distance_To_Roadways, Hillshade_9am, Hillshade_Noon
Wilderness_Area4, Soil_Type1, Soil_Type2, Soil_Type3, Soil_Type4, Soil_Type5, Soil_Type6,
Soil_Type9, Soil_Type10, Soil_Type11, Soil_Type12, Soil_Type13, Soil_Type16, Soil_Type17,
Soil_Type18, Soil_Type19, Soil_Type20, Soil_Type22, Soil_Type23, Soil_Type24,
```

```
Soil_Type26, Soil_Type27, Soil_Type29, Soil_Type30, Soil_Type31, Soil_Type32, Soil_Type33,
Soil_Type34, Soil_Type35, Soil_Type38, Soil_Type39, Soil_Type40, Cover_Type,
Square_Hypo_Distance, Average_Dist_Road_Hydro, Average_Elevation_Hydro
FROM covtype1;
```

'Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology', 'Vertical_Distance_To_Hydrology', 'Horizontal_Distance_To_Roadways', 'Hillshade_9am', 'Hillshade_Noon', 'Hillshade_3pm', 'Horizontal_Distance_To_Fire_Points',

- Exporting transformed dataset (`covtype1`) from SQLite to local PC as `csv` file :

1.1.3 Let's move to the second file (`02_ForestProject-Modeling&Presentation`) for modeling and presentation.

```
[ ]:
```