FOREST COVERTYPE CLASSIFICATION

Aim:

To classify the forest covertype based on the attributes obtained from Department of Forest Services in US. Forest Covertype is of 7 types and hence it is a multilabel classification.

Dataset:

Source of the Dataset: https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/

Number of instances(observations: 581012

Number of Attributes: 12 measures, but 54 columns of data(10 quantitative variables, 4 binary wilderness areas and 40 binary soil type variables)

Metadata of the Dataset:

Name	**Measurement**
Elevation	meters
Aspect	azimuth
Slope	degrees
Horizontal_Distance_To_Hydrology	meters
Vertical_Distance_To_Hydrology	meters
Horizontal_Distance_To_Roadways	meters
Hillshade_9am	0 to 255 index
Hillshade_Noon	0 to 255 index
Hillshade_3pm	0 to 255 index
Horizontal_Distance_To_Fire_Points	Meters
Wilderness_Area (4 binary columns)	0(absence) or 1 (presence)
Soil_Type (40 binary columns)	0(absence) or 1 (presence)
Cover_Type (7 types)	1 to 7

Forest Cover Type Classes: 1 -- Spruce/Fir

2 -- Lodgepole Pine

3 -- Ponderosa Pine

4 -- Cottonwood/Willow

5 -- Aspen

6 -- Douglas-fir

7 – Krummholz

Sampling of Dataset

The dataset contains 581012 obervations and 55 attributes.

Random sampling is performed on dataset and obtained 200000 samples.

```
dataset=df.sample(200000,random_state=1)
```

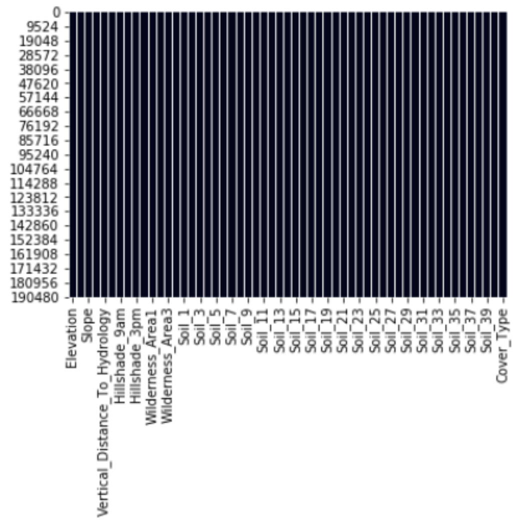
CoverType_sampling.ipynb contains the logic for sampling the observations.

Exploratory Data Analysis:

Check for missing values:

cover is the dataset variable name used in the entire code.

The heatmap of the dataset indicates that there are no null values in the dataset. If there are null values we have to perform imputation techniques and if more null values we need to drop that columns.



Compare the correlation values and drop the column

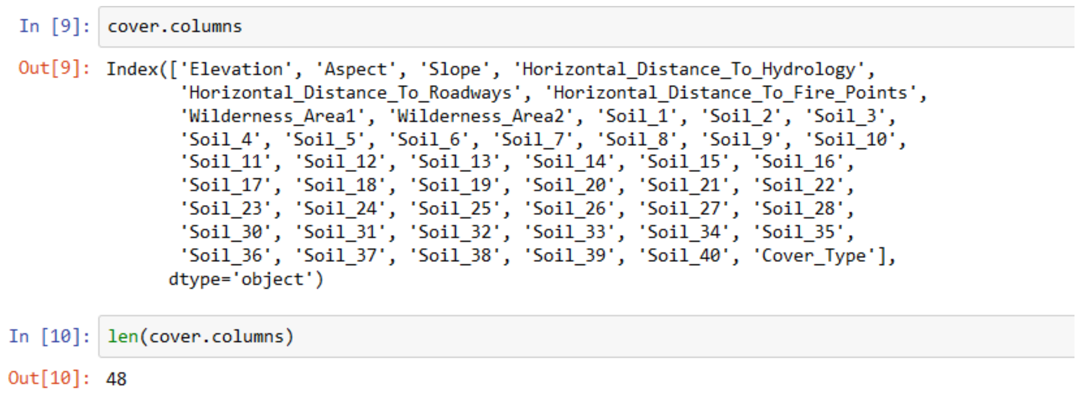
We are checking for the correlation between columns and drop one of the columns which is highly correlated with other column.

In general, Correlation is used to summarize the strength and direction of the linear association between two quantitative variables

Condition used in the code:

If the correlation value between x and y columns is greater than 0.5, drop x.

After the correlation drop technique, the number of columns reduced from 55 to 48.



```
In [9]: cover.columns

Out[9]: Index(['Elevation', 'Aspect', 'Slope', 'Horizontal_Distance_To_Hydrology',
               'Horizontal_Distance_To_Roadways', 'Horizontal_Distance_To_Fire_Points',
               'Wilderness_Area1', 'Wilderness_Area2', 'Soil_1', 'Soil_2', 'Soil_3',
               'Soil_4', 'Soil_5', 'Soil_6', 'Soil_7', 'Soil_8', 'Soil_9', 'Soil_10',
               'Soil_11', 'Soil_12', 'Soil_13', 'Soil_14', 'Soil_15', 'Soil_16',
               'Soil_17', 'Soil_18', 'Soil_19', 'Soil_20', 'Soil_21', 'Soil_22',
               'Soil_23', 'Soil_24', 'Soil_25', 'Soil_26', 'Soil_27', 'Soil_28',
               'Soil_30', 'Soil_31', 'Soil_32', 'Soil_33', 'Soil_34', 'Soil_35',
               'Soil_36', 'Soil_37', 'Soil_38', 'Soil_39', 'Soil_40', 'Cover_Type'],
              dtype='object')

In [10]: len(cover.columns)

Out[10]: 48
```

Feature Selection Using Chi Square Distribution

Chi square distribution is used to measure the degree of association between two categorical variables

Chi square uses hypothesis testing.

- H0: The attribute X has no role to play in the forest covertype (The feature is not important)
- H1: The attribute X has a role to play in forest covertype (The feature is important)

X represents all the independent variables.

chi_scores=chi2(X,y)

#chi_cores yields the chi square statistic and p-values

Higher the chi square value indicates to reject the H0 and the feature is more important. The last 15 columns which are having less chi square value are dropped

Feature Selection Using Anova Test

ANOVA is a popular feature selection techniques that can be used for numerical input data and a categorical (class) target variable. ANOVA is an acronym for “analysis of variance” and is a parametric statistical hypothesis test for determining whether the means from two or more samples of data (often three or more) come from the same distribution or not.

ANOVA assumes the below hypothesis

H0: Means of all groups are equal.

H1: At least one mean of the groups are different.

aov_table = sm.stats.anova_lm(mod, typ=2)

	sum_sq	df	F	PR(>F)
Elevation	19128.638038	1.0	10802.836162	0.000000e+00
Aspect	78.131693	1.0	44.124619	3.089056e-11
Slope	1763.777815	1.0	996.087788	4.391222e-218
Horizontal_Distance_To_Hydrology	1352.292526	1.0	763.702809	8.721898e-168
Horizontal_Distance_To_Roadways	335.276628	1.0	189.346386	4.620176e-43
Horizontal_Distance_To_Fire_Points	764.415326	1.0	431.701071	8.752853e-96
Residual	354128.642670	199993.0	NaN	NaN

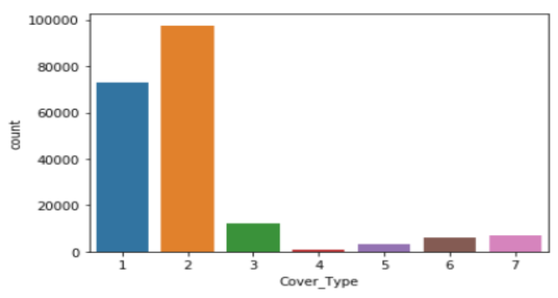
Pr > F – This is the p-value associated with the F statistic of a given source.

If PR>F is greater than 0.05, we have to remove that column. But in our case all continuous variables has less than 0.05 value, hence we are not dropping any columns.

Data Visualization

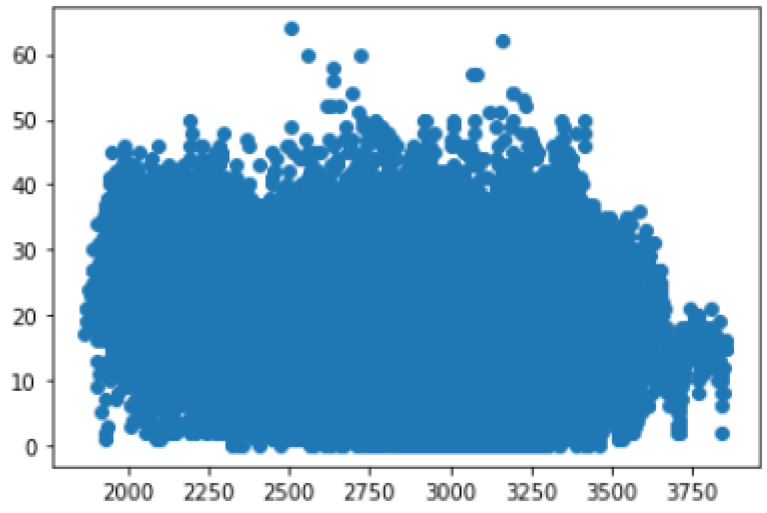
sns.countplot(cover['Cover_Type'])

```
<matplotlib.axes._subplots.AxesSubplot at 0x146b1b2a648>
```



There are 7 classes in target variable, Class 1 and 2 covers most of the data distribution plt.scatter(cover['Elevation'],cover['Slope'])

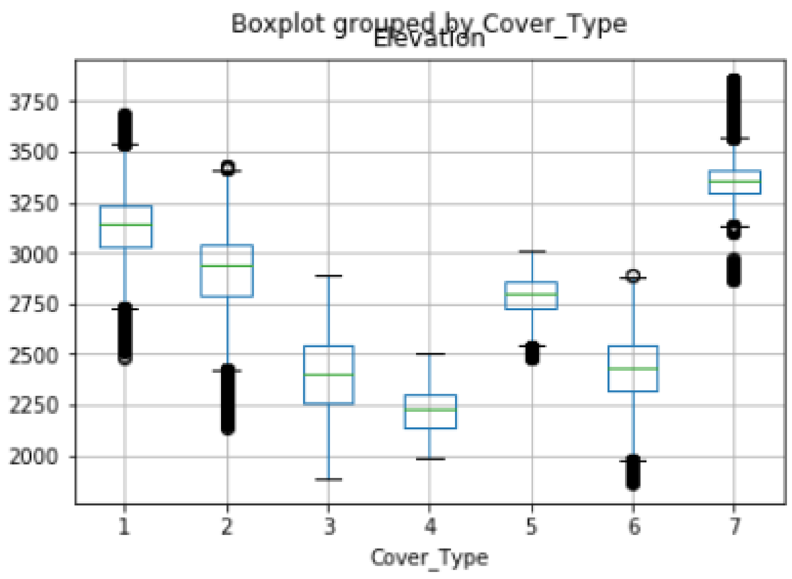
```
<matplotlib.collections.PathCollection at 0x1d8835a07c8>
```



Inference: We could not find any correlation between Elevation and Slope

cover.boxplot(by='Cover_Type',column=['Elevation'])

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d8839a5208>
```



Inference: Compared the distribution of Elevation with Cover_Type to understand if the interval of Elevation values directly matches with the Cover_type but there are overlapping intervals

Splitting the input features and target variable

x=~Cover_Type (All columns except Cover_Type)

y=Cover_Type

Train Test split

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.25,random_state=42)

train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. Train_test_split() uses the following parameters.

- arrays — the dataset to be split;
- test_size — the size of the test set.
- train_size — the size of the train set. Its behavior is complementary to the test_size variable.
- random_state — before applying to split, the dataset is shuffled. The random_state variable is an integer that initializes the seed used for shuffling. It is used to make the experiment reproducible.

CLASSIFICATION MODEL

Random Forest Classifier is used in this dataset.

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

```
ACCURACY OF THE MODEL:  0.93726
```
Classification Report:

```
               precision    recall  f1-score   support

           1       0.95      0.93      0.94     18138
           2       0.93      0.96      0.95     24327
           3       0.91      0.94      0.92      3145
           4       0.89      0.83      0.86       235
           5       0.93      0.72      0.81       825
           6       0.90      0.84      0.87      1575
           7       0.97      0.91      0.94      1755

    accuracy                           0.94     50000
   macro avg       0.93      0.88      0.90     50000
weighted avg       0.94      0.94      0.94     50000
```

We could see the class 4,5,6 having less recall and f1-score which is due to the data imbalance in the dataset. This is can be resolved by **class weight technique.**

Class Distribution:

```
1    54768
2    73271
3     9166
7     5260
6     4418
5     2410
4      707
```

Class weights modify the loss function directly by giving a penalty to the classes with different weights. It means purposely increasing the power of the minority class and reducing the power of the majority class.

clf = RandomForestClassifier(n_estimators = 100,class_weight='balanced')

The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y))

Classification Report:

```
                precision    recall  f1-score   support

           1       0.95      0.93      0.94     18138
           2       0.94      0.96      0.95     24327
           3       0.91      0.94      0.93      3145
           4       0.89      0.84      0.86       235
           5       0.93      0.71      0.80       825
           6       0.89      0.84      0.87      1575
           7       0.97      0.91      0.94      1755

    accuracy                           0.94     50000
   macro avg       0.93      0.88      0.90     50000
weighted avg       0.94      0.94      0.94     50000
```

There is no much increase in the recall value of the classes. There is a slight increase in the recall value of Class 5

To improve further, we can focus more on

- Feature Engineering
- Turning the threshold of algorithm
- Using advanced algorithms