Individual Project Report

Forest Cover Type prediction

Contributor: Yashwant Bhaidkar

Introduction:

The goal of this project is to predict the forest cover type (one of the seven types) based on the given attributes. It has been used in numerous studies and competitions to evaluate the performance of various machine-learning algorithms and techniques.

UCI forest cover prediction dataset: This dataset is imbalanced and it has around 581012 data points. It was created in the GIS Program at Colorado State University. The dataset was obtained from the UCI repository.

The dataset consists of 581,012 instances with 54 attributes, The quantitative variables include elevation, slope, aspect, horizontal distance to water, and other terrain variables. The binary variables include wilderness area, soil type, and tree type.

In this dataset we have,

- 10 Numerical features
- 44 Categorical features

Description of individual Contribution:

- Performed on the EDA part for categorical features.
- Performed standard scaling and normalize skewed and bimodal features.
- Worked on the feature engineering part of the soil features and created 7 new features based on soil order type
- Performed feature engineering using RFE
- Developed Baseline model(Logistic regression for Multiclass classification)
- Tuned RF model and built the best RF model based on hyperparameter tuning.

Work Description and Results:

EDA:

I worked on some of the numerical features and categorical features:

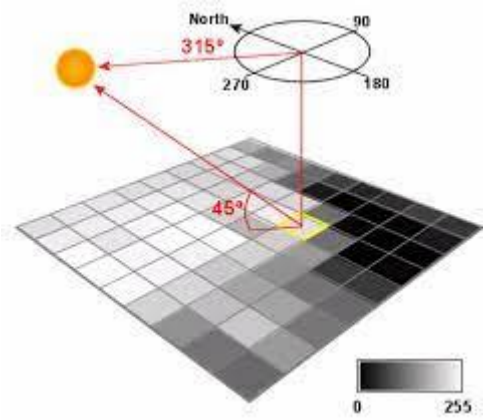
There is no missing information (Null) in this dataset.

Some of the features which I covered

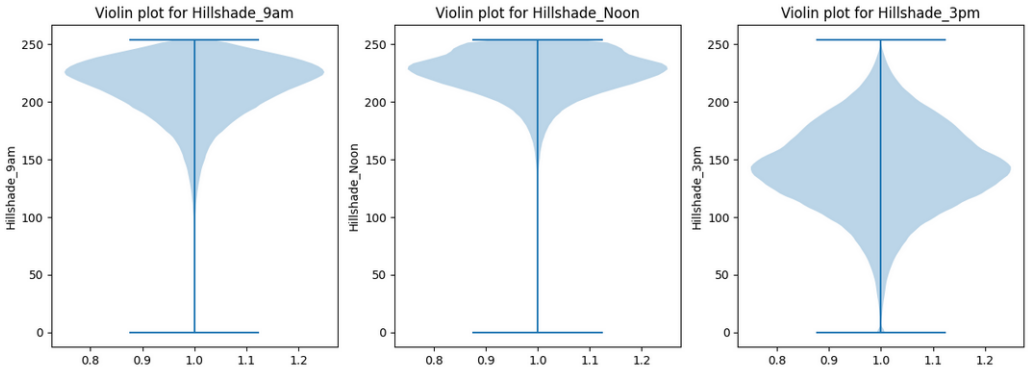
Hill shade features:

We have 3 features based on the hill shade values for 3 particular times during the day.

This feature is important as it will give us an idea that how much sunlight is available at that spot.



These values range from 0 to 255, so it is giving us the color shade between black and white. (0 is dark and 255 is white)

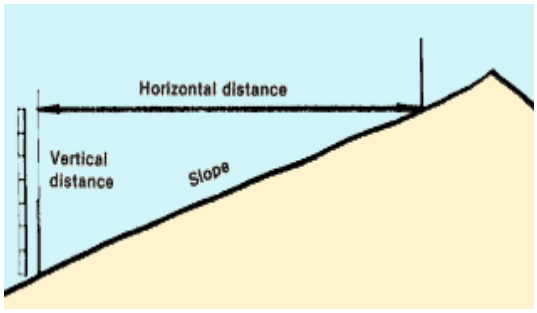


Based on the violin plot, hill shade 3 am looks normal and hillshade_9AM and Hillshade_Noon are left skewed. As the range is fixed, we can say that there is no outlier present.

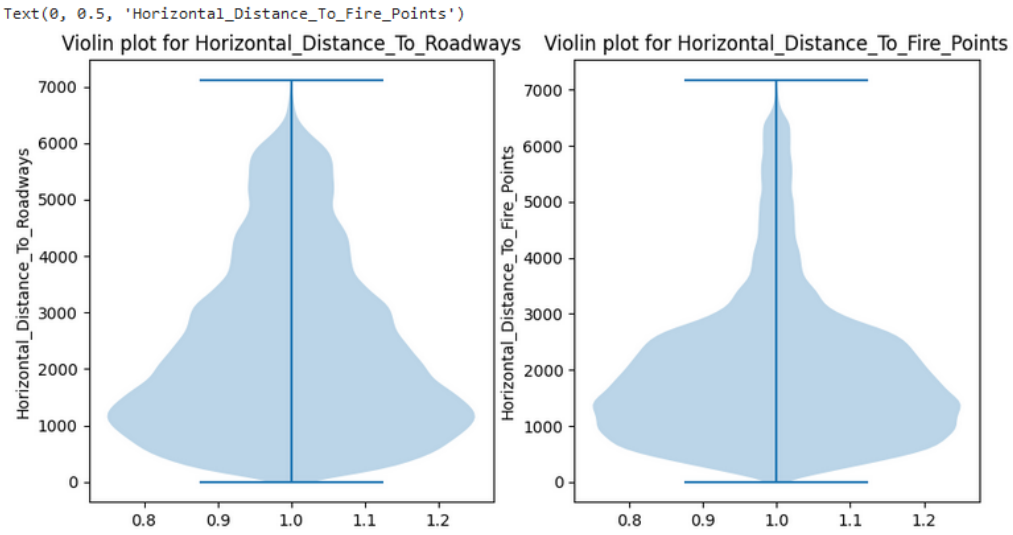
Horizontal distance to roadways and fire points:

Horizontal distance is the distance of the survey spot from the roadways and fire points.

The main impact of this feature is on the forest quality as if the locations are away from the roads and fire points, those spots are more secure and we will get better quality of forest cover as there will be less pollution and the probability of damage from the fire will be very less.



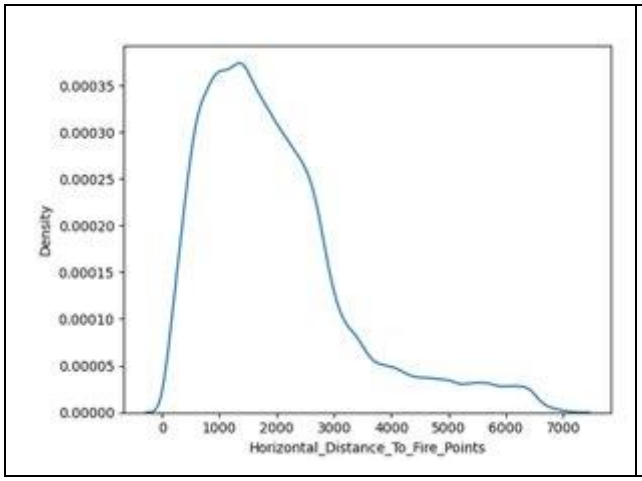
This feature is explaining the horizontal and vertical distance from the survey location to the roadways



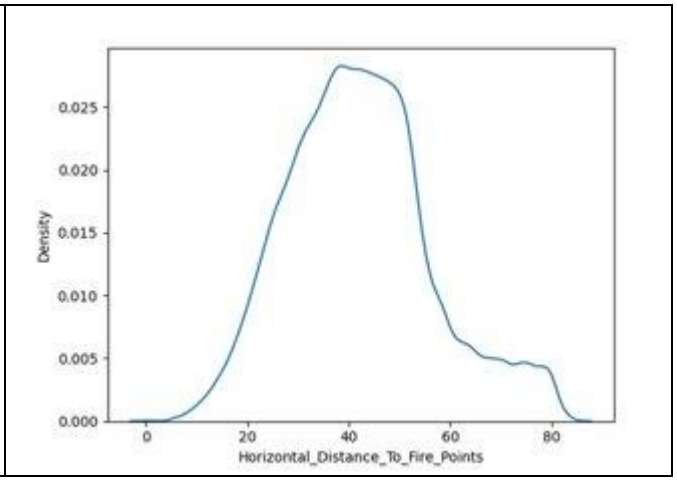
We can clearly see that these features are heavily right-skewed and the value range we have is from 0 to 7000.

To normalize that, we can apply log on the feature as it will shift the higher range values towards the left and we will get an approximate normal distribution.

Feature Engineering on Numerical data:

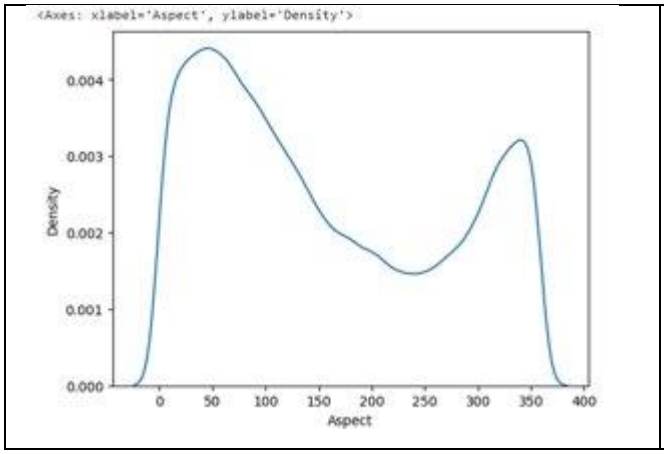


Right Skewed feature

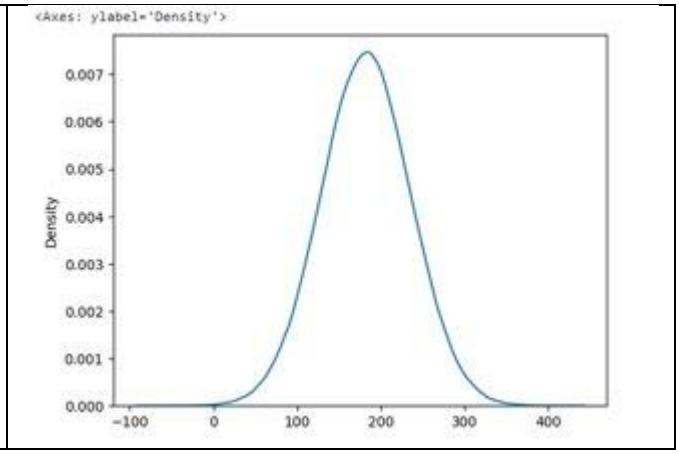


Normalized feature using SQRT transform

Normalized the feature which was right skewed to approximate normal feature using sqrt transform.



Bimodal Feature



Normalized feature using Gaussian mixture

Standard Scaling:

As some of the features are nearly normal and so performed standard scaling on it.

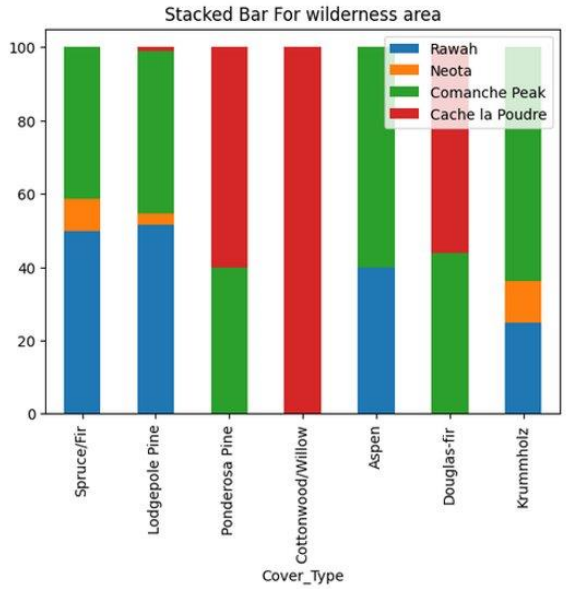
```
Run Cell | Run Above | Debug Cell
# %%
from sklearn.preprocessing import StandardScaler
def standard_scaling(dfa,column_list):
    for i in column_list:
        tf = StandardScaler()
        dfa[i] = tf.fit_transform(dfa[i].values.reshape(-1,1))
    return dfa
```

This function is taking dataframe and column names and provides scaled featured dataframe for the features we mentioned in the column list parameters.

Wilderness area:

We have 4 features based on wilderness area type.

To find the impact of these features on the cover prediction, we mapped the percentage wise class of each wilderness area for particular class type(cover type).



- Based on this graph we can say that, Cache la Paudre has mostly cottonwood/Willow cover type.
- Most of the cover types are observed in wilderness area 3(Comanche Peak)

- Very small percentage of the cover type was observed in Neota.
- Cover types 1,2,5 and 7 were observed in Rawah.

Feature Engineering:

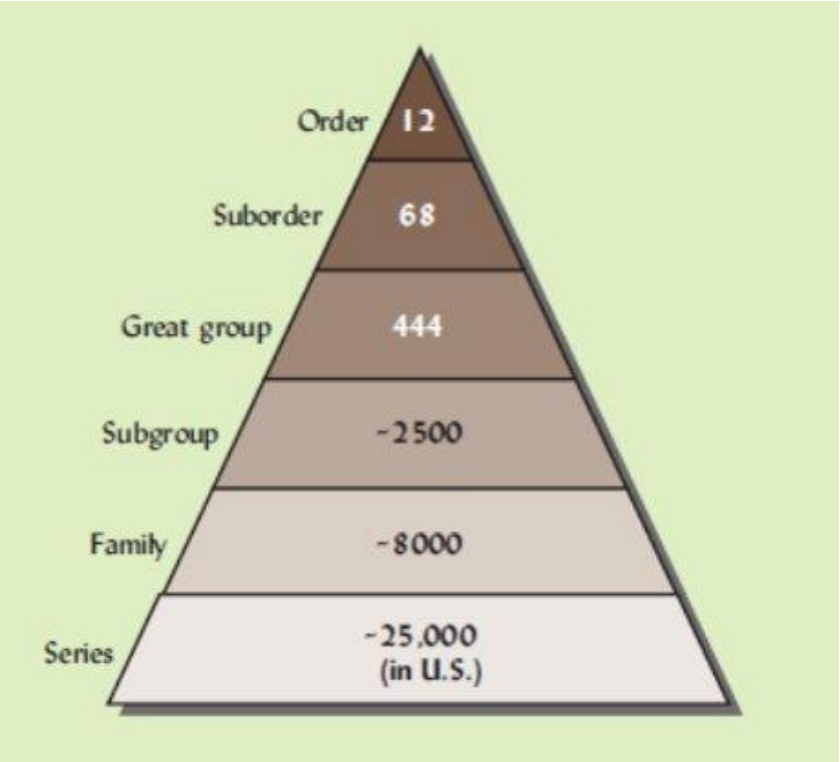
Soil Features:

we have 40 different soil types with the soil texture description.

Soil	Family	Description
Soil_Type2	Vanet	Ratake families complex, very stony.
Soil_Type5	Vanet	Rock outcrop complex complex, rubbly.
Soil_Type6	Vanet	Wetmore families - Rock outcrop complex, stony.

As we observed that some of the soil descriptions are common, there must be some similarity in these soil types and we can find out the strong relations which are representing the given types into strong groups.

Soil	Family	Description	Subgroup	Order
Soil_Type2	Vanet	Ratake families complex, very stony.	Calcidic Haplustalfs	Alfisols
Soil_Type5	Vanet	Rock outcrop complex complex, rubbly.	Calcidic Haplustalfs	Alfisols
Soil_Type6	Vanet	Wetmore families - Rock outcrop complex, stony.	Calcidic Haplustalfs	Alfisols



In case of soil classification, there are 12 main orders of the soil types.

Order – Twelve soil orders are recognized. The differences among orders reflect the dominant soil forming processes and the degree of soil formation. Each order is identified by a word ending in 'sol.' An example is Alfisols.

Suborder - Each order is divided into suborders primarily on the basis of properties that influence soil formation and/or are important to plant growth.

Great Group – Each suborder is divided into great groups on the basis of similarities in horizons present, soil moisture or temperature regimes, or other significant soil properties.

Subgroup – Each great group has a 'typic' (typical) subgroup which is basically defined by the Great Group. Other Subgroups are transitions to other orders, suborders, or great groups due to properties that distinguish it from the great group.

Family – Families are established within a subgroup on the basis of physical and chemical properties along with other characteristics that affect management.

Series – The series consists of soils within a family that have horizons similar in color, texture, structure, reaction, consistence, mineral and chemical composition, and arrangement in the profile.

The main 12 Soil orders are **Entisols, Inceptisols, Andisols, Mollisols, Alfisols, Spodosols, Ultisols, Oxisols, Gelisols, Histosols, Aridisols, and Vertisols**.

Each order is based on one or two dominant physical, chemical, or biological properties that differentiate it clearly from the other orders. Perhaps the easiest way to understand why certain properties were chosen over others is to consider how the soil (i.e., land) will be used.

As we have 40 different soil types in the dataset, it is really hard to manage this many features in the model and we can reduce these features by classifying them into main parent classes.

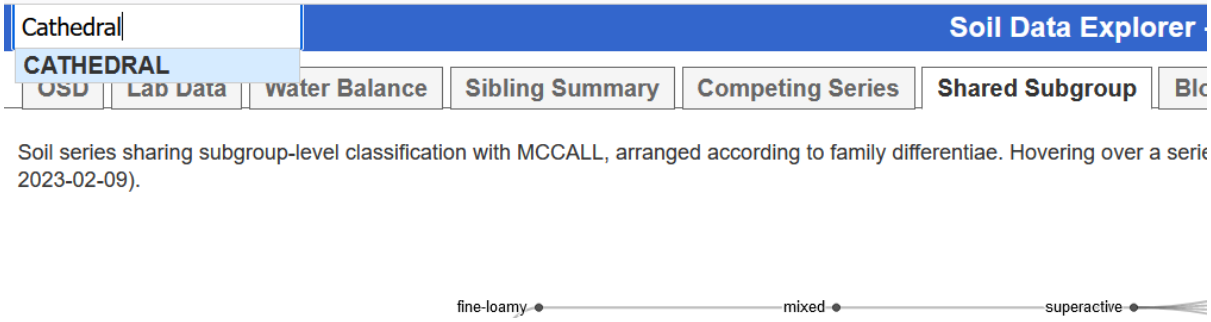
With the help of various survey sites like soilweb.com, we can easily find out the parent type of the given soil family. We can easily find out the order and sub-group with the help of survey information.

Example:

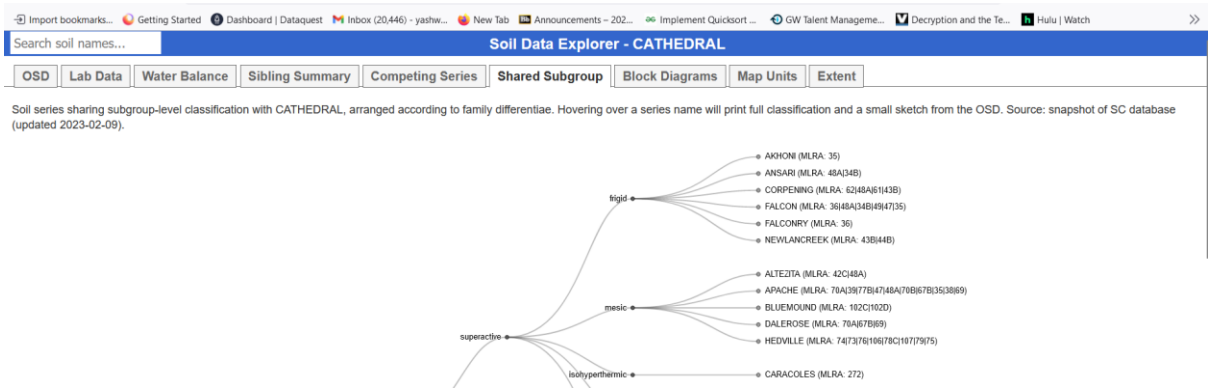
Soil_Type1 is from the Cathedral family.

search the family in

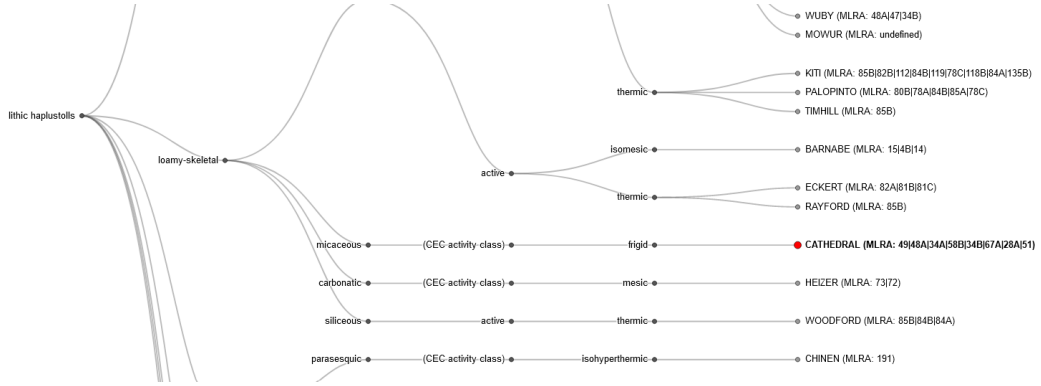
https://soilmap2-1.lawr.ucdavis.edu/sde/?series=mccall#shared-subgroup



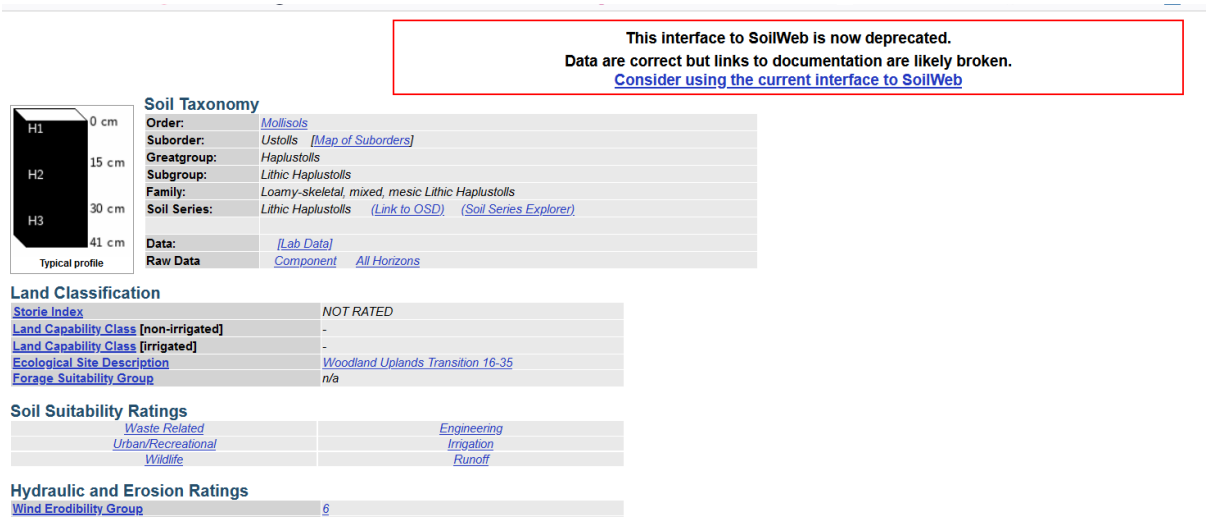
Go to the Shared subgroup option for the hierarchical diagram



In this tab we will get the main parent sub group of the soil family.



Go to soil website and search for sub-group. We will get to see the main order of the soil.



This interface to SoilWeb is now deprecated.
Data are correct but links to documentation are likely broken.
Consider using the current interface to SoilWeb

H1 0 cm
H2 15 cm
H3 30 cm
41 cm
Typical profile

Soil Taxonomy

Order:	*Mollisols*
Suborder:	*Ustolls* [Map of Suborders]
Greatgroup:	*Haplustolls*
Subgroup:	*Lithic Haplustolls*
Family:	*Loamy-skeletal, mixed, mesic Lithic Haplustolls*
Soil Series:	*Lithic Haplustolls* (Link to OSD) (Soil Series Explorer)
Data:	[Lab Data]
Raw Data	Component All Horizons

Land Classification

Storie Index	*NOT RATED*
Land Capability Class [non-irrigated]	-
Land Capability Class [irrigated]	-
Ecological Site Description	*Woodland Uplands Transition 16-35*
Forage Suitability Group	*n/a*

Soil Suitability Ratings

Waste Related	Engineering
Urban/Recreational	Irrigation
Wildlife	Runoff

Hydraulic and Erosion Ratings

Wind Erodibility Group	6

Next, we will check if there is any strong relationship between soil order types and forest cover types.

Soil	Family	Description	Subgroup	Order
Soil_Type2	Vanet	Ratake families complex, very stony.	Calcidic Haplustalfs	Alfisols
Soil_Type5	Vanet	Rock outcrop complex complex, rubbly.	Calcidic Haplustalfs	Alfisols
Soil_Type6	Vanet	Wetmore families - Rock outcrop complex, stony.	Calcidic Haplustalfs	Alfisols

This is how we got the mapping of 40 soil types into 6 major soil types. There is one type for which we do not have any information hence we categorize that as unknown.

Using the soil order mapping, classifying 40 soil types into 7 main soil orders.

```
[ ] for i in range(0, len(df)):
      mol = ['Soil_Type1','Soil_Type3','Soil_Type4','Soil_Type7','Soil_Type8','Soil_Type1','Soil_Type14','Soil_Type16','Soil_Type17','Soil_Type18']
      alf = ['Soil_Type2','Soil_Type5','Soil_Type6','Soil_Type9','Soil_Type26']
      ent = ['Soil_Type12','Soil_Type34']
      hist = ['Soil_Type19']
      enc = ['Soil_Type10','Soil_Type11','Soil_Type32','Soil_Type28','Soil_Type13','Soil_Type20','Soil_Type21','Soil_Type22','Soil_Type23','Soil_Type24','Soil_Type33','Soil_Type27','Soil_Type25','Soil_Type38','Soil_Type31','Soil_Type29','Soil_Type30']
      spod = ['Soil_Type35','Soil_Type36','Soil_Type37','Soil_Type39','Soil_Type40']
      Unknown = ['Soil_Type15']
      flag = 0

      if i%50000 == 0:
        print(i)

      for x1 in mol:
        if df[x1][i]== 1:
          df['Mollisols'][i] = 1
          flag = 1
          break
```

We are using the above method to map the order types to the respective soil type.

Important soil types based on predicted forest classes:

As per the soil information, it is observed that we have some soil types which are contributing more in the prediction of a particular soil type.

So, we find out the percentage-wise contribution of each soil type in each class prediction.

```
[ ] soil_contri
```

	Cover_Type	Soil_Type1	Soil_Type2	Soil_Type3	Soil_Type4	Soil_Type5	Soil_Type6	Soil_Type7	Soil_Type8	Soil_Type9	Soil_Type10	Soil_Type11	Soil_Type12	Soil_Type13	Soil_Type14	Soil_Type15	Soil_Type16	Soil_Type17
0	1	NaN	NaN	NaN	0.085914	NaN	NaN	NaN	0.020298	0.076001	0.451284	0.352625	1.271242	1.037103	NaN	NaN	0.300227	0.101020
1	2	NaN	0.300740	0.420401	1.147543	NaN	0.321919	0.037063	0.048005	0.348040	3.813259	3.204013	9.628628	4.679828	NaN	NaN	0.615247	0.337803
2	3	5.876266	13.959277	6.743301	20.979471	2.704592	11.167981	NaN	NaN	NaN	32.253734	3.784192	NaN	0.114672	0.324439	NaN	0.360799	1.415226
3	4	6.479796	4.186385	37.058609	6.115763	1.747361	11.649072	NaN	NaN	NaN	8.154350	1.237714	NaN	NaN	5.642519	NaN	1.856571	15.871860
4	5	NaN	2.780997	NaN	6.162435	NaN	NaN	NaN	NaN	NaN	2.738860	7.173707	NaN	13.852312	NaN	NaN	0.368693	6.320447
5	6	4.330051	7.502735	1.168884	3.633328	3.351183	7.773363	NaN	NaN	NaN	51.010537	2.982668	NaN	3.535441	1.888639	0.017274	1.445270	4.082455
6	7	NaN	NaN	NaN	0.380302	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.029254	NaN	NaN	NaN	NaN

There are some soil types, which are helpful in predicting the forest cover type and we will include those features in our model

Based on these observations, we can say that Soil_Type29, Soil_Type10, Soil_Type3, Soil_Type30 and Soil_Type38 are important in cover prediction.

Feature Selection:

Used RFE to get the feature importance with random forest.
Random forest works really well on imbalanced data and it uses entropy or gini to evaluate the node and split it based on information gain.
So if the feature is used multiple times to split the nodes, that feature will be the important feature.

```
tf = RFE(RandomForestClassifier(), n_features_to_select=30, verbose=1)
Xt = tf.fit_transform(x_train,y_train)
print("Shape =", Xt.shape)
```

```
Fitting estimator with 61 features.
```

We got the top 30 features using get_feature_names_out() method.

```
[ ] tf.get_feature_names_out()
```

```
array(['Elevation', 'Slope', 'Horizontal_Distance_To_Hydrology',
       'Vertical_Distance_To_Hydrology',
       'Horizontal_Distance_To_Roadways', 'Hillshade_9am',
       'Hillshade_Noon', 'Hillshade_3pm',
       'Horizontal_Distance_To_Fire_Points', 'Wilderness_Area1',
       'Wilderness_Area2', 'Wilderness_Area3', 'Wilderness_Area4',
       'Soil_Type2', 'Soil_Type4', 'Soil_Type10', 'Soil_Type12',
       'Soil_Type22', 'Soil_Type23', 'Soil_Type24', 'Soil_Type29',
       'Soil_Type32', 'Soil_Type33', 'Soil_Type38', 'Inceptisols',
       'Mollisols', 'Spodosols', 'Alfisols', 'Entisols',
       'Aspect_unimodal'], dtype=object)
```

Data Splitting:

To perform the hyperparameter tuning, we used the CV data and checked the performance of the model firstly on CV and then then we checked the test output of each model.

```
x_train,x_test,y_train,y_test = split_data(top_30_features,'Cover_Type',0.3)
```

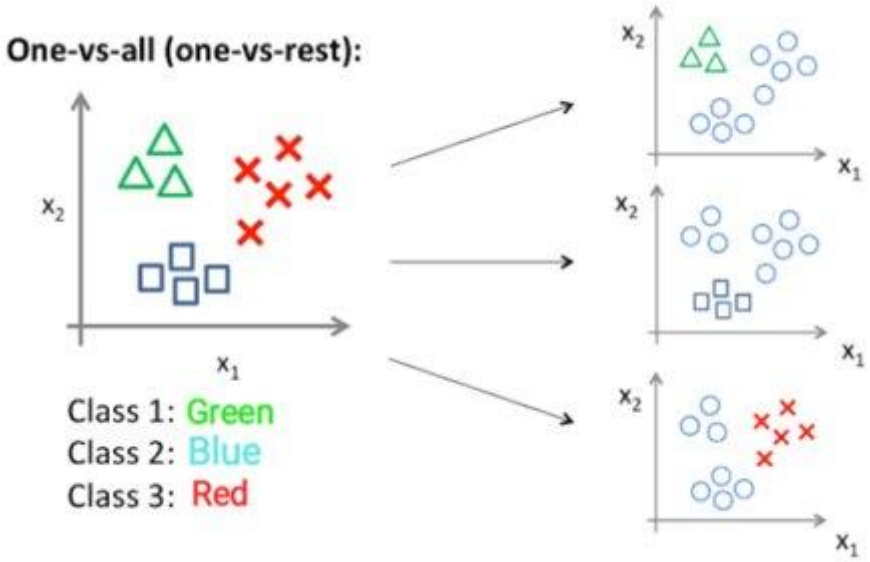
```
x_train,x_cv,y_train,y_cv = train_test_split(x_train,y_train,test_size = 0.2)
```

Model Building:

Base model:

Logistic Regression for Multiclass:

Used the logistic regression using one vs all approach for multi-class.
It will create sub-models for each class and provides the output based on the higher probability of the class.



This model builds the sub-models internally for each class and it creates the fitting line for each class by considering it as 1 and the other classes as 0.
When it gets the data point for prediction, it tries to fit that data point based on all sub-models and checks the probability of each class. The winning class will have the highest probability.

	precision	recall	f1-score	support
1	0.71	0.69	0.70	63717
2	0.75	0.80	0.77	84934
3	0.67	0.80	0.73	10620
4	0.54	0.35	0.43	825
5	0.45	0.04	0.08	2842
6	0.49	0.26	0.34	5230
7	0.73	0.57	0.64	6136
accuracy			0.72	174304
macro avg	0.62	0.50	0.53	174304
weighted avg	0.71	0.72	0.71	174304

Here, we got the average model as we are using the base features only. F1 macro score is 0.53 which is very average. If we check the F1 score of the individual class, the F1 score is not that good.

Random forest with hyperparameter tuning:

As grid search and randomized cv search were taking too long, we decided to perform the hyperparameter tuning using for loop.

We passed the depth and number of estimators in for loop and check the F1 score for each model using CV and Test data.

Hyperparameter tuning using for loop

```
[ ] from sklearn.metrics import f1_score
    depth = [3,5,10,20,50,100]
    cv_f1_score = []
    train_f1_score = []
    for i in depth:
      model = RandomForestClassifier(max_depth = i,n_jobs = -1,n_estimators = 10)
      model.fit(X_train_smote, Y_train_smote)
      CV = CalibratedClassifierCV(model,method = 'sigmoid')
      CV.fit(X_train_smote, Y_train_smote)
      predicted = CV.predict(x_cv)
      train_predicted = CV.predict(x_train)
      cv_f1_score.append(f1_score(y_cv,predicted,average = 'macro'))
      train_f1_score.append(f1_score(y_train,train_predicted,average = 'macro'))
      print('depth {0} is finished'.format(i))
    for i in range(0,len(cv_f1_score)):
      print('f1 value score for depth =' + str(depth[i]) + ' is ' + str(cv_f1_score[i]))
    plt.plot(depth,cv_f1_score,c='r')
    plt.plot(depth,train_f1_score,c='b')
    plt.xlabel('depth(depth of the tree)')
    plt.ylabel('f1 score(train and test)')
```

As grid search was taking too long to get the output (the dataset is huge), we decided to go with the for loop based hyperparameter tuning, where we are passing the list of hyperparameters and checked the CV and test data F1 score.

```
[ ]  pred=best_rf_model.predict(x_test)
     print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	63548
1	0.96	0.96	0.96	84956
2	0.93	0.95	0.94	10711
3	0.86	0.89	0.88	813
4	0.81	0.92	0.86	2846
5	0.85	0.92	0.89	5187
6	0.94	0.98	0.96	6243
accuracy			0.95	174304
macro avg	0.90	0.94	0.92	174304
weighted avg	0.95	0.95	0.95	174304

We got an F1 score of around .92 which is a stable model. Precision and recall are also matching for each class and the model is trying to balance the same.

Summary and conclusions:

- The bagging algorithm really works well and there is a good balance between Precision and recall.
- Soil order type features are contributing to the top 30 features.
- Feature engineering and hyperparameter tuning improved results for the Random Forest Classifier and XGBoost Classifier by 1% and 1.2% respectively.
- Some soil features are contributing to the cover type prediction with a good margin.
- Some other features like weather conditions and geographical information like location could help us in improving our results further.

Code Percentage:

Code copied: 150 lines
Modified code: 50
Self-written code: around 450 lines

(150 – 50)/(450 + 150) = 100/600 = 17%

References:

http://archive.ics.uci.edu/ml/datasets/covertype

https://soilmap2-1.lawr.ucdavis.edu/sde/?series=mccall#shared-subgroup

https://digitalatlas.cose.isu.edu/geo/soils/soiltxt/soiltax.htm

https://rstudio-pubs-static.s3.amazonaws.com/160297_f7bcb8d140b74bd19b758eb328344908.html

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html