Deliverable 3

Yuyan Chen

March 2021

1 Final Training Results

I used LinearSVC and KNN for Deliverable2.
The best result I got from LinearSVC was 73.72% training accuracy and 58.16 % testing accuracy. The best result I got from KNN with 5 neighbors was 95.92% training accuracy and 59.83 % testing accuracy. Both models were overfitting.

1.1 Data Preprocessing

There were several problems in the previous data preprocessing, and I changed the process in the following way:

1. I removed two features: Horizontal_Distance_To_Hydrology and Vertical_Distance_To_Hydrology and added Distance_To_Hydrology instead. Distance_To_Hydrology was calculated as

   $$Distance = \sqrt{Horizontal\_Distance^2 + Vertical\_Distance^2}$$

   This new feature was more informative than the previous two features.

2. I used train_test_split to spilt with shuffling to split the dataset into training, validation, and testing set. For Deliverable 2, I simply used first 371,848 records for training, next 92,962 records for validation, and last 116,202 records for testing. Since adjacent data points have strong correlations, my previous splitting method introduced a bias in the data.

3. I used naive random oversampling to create a balanced training set, which increased both the accuracy and balanced accuracy of the models.

1.2 Change of Metrics

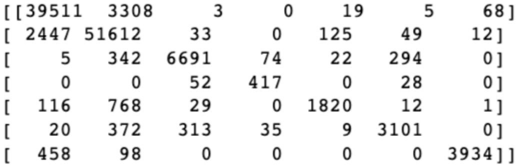
I changed the metric from accuracy to balanced accuracy and confusion matrix, as my dataset is highly imbalanced and accuracy can be misleading.

1.3 Results

1.3.1 KNN

I used KNN with 5 neighbors. The balanced testing accuracy is 90.79%, much higher than last time. However, KNN was a very slow method as it reviews

Figure 1.1: Confusion matrix of testing set



```
[[39511  3308     3     0    19     5    68]
 [ 2447 51612    33     0   125    49    12]
 [    5   342  6691    74    22   294     0]
 [    0     0    52   417     0    28     0]
 [  116   768    29     0  1820    12     1]
 [   20   372   313    35     9  3101     0]
 [  458    98     0     0     0     0  3934]]
```

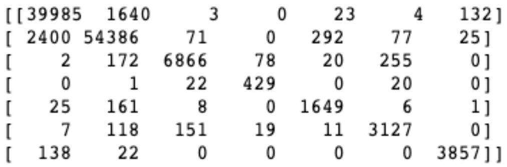
the entire dataset again to make a prediction. It took more than 3 hours to predict the training test with 1,270,115 data points (https://github.com/Yuyan-C/MAIS202Project/blob/main/newKNN.ipynb).

1.3.2 Random Forest

I used sklearn.ensemble.RandomForestClassifier with default parameters. The result is surprising:

- training accuracy (balanced): 0.9999992126697189
- validation accuracy (balanced): 0.9144780575065823
- testing accuracy (balanced): 0.9093409409040784

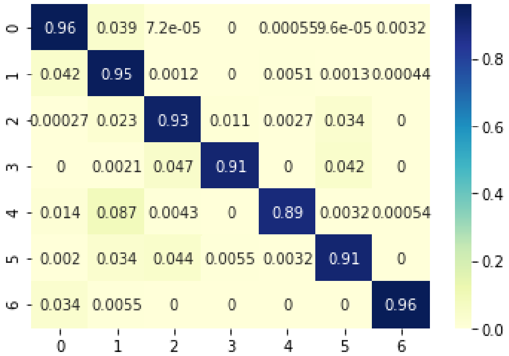
Figure 1.2: Confusion matrix of testing set



```
[[39985  1640     3     0    23     4   132]
 [ 2400 54386    71     0   292    77    25]
 [    2   172  6866    78    20   255     0]
 [    0     1    22   429     0    20     0]
 [   25   161     8     0  1649     6     1]
 [    7   118   151    19    11  3127     0]
 [  138    22     0     0     0     0  3857]]
```

This is the visualization of the confusion matrix. Since the testing set is imbalanced, the confusion matrix is plotted with percentage.

Figure 1.3: Heatmap of the confusion matrix (percentage)



	0	1	2	3	4	5	6
0	0.96	0.039	7.2e-05	0	0.00055	9.6e-05	0.0032
1	0.042	0.95	0.0012	0	0.0051	0.0013	0.00044
2	0.00027	0.023	0.93	0.011	0.0027	0.034	0
3	0	0.0021	0.047	0.91	0	0.042	0
4	0.014	0.087	0.0043	0	0.89	0.0032	0.00054
5	0.002	0.034	0.044	0.0055	0.0032	0.91	0
6	0.034	0.0055	0	0	0	0	0.96

Random Forest classifier also makes prediction much faster than the KNN classifier. Therefore, I choose this model as my final result (https://github.com/Yuyan-C/MAIS202Project/blob/main/RandomForest.ipynb).

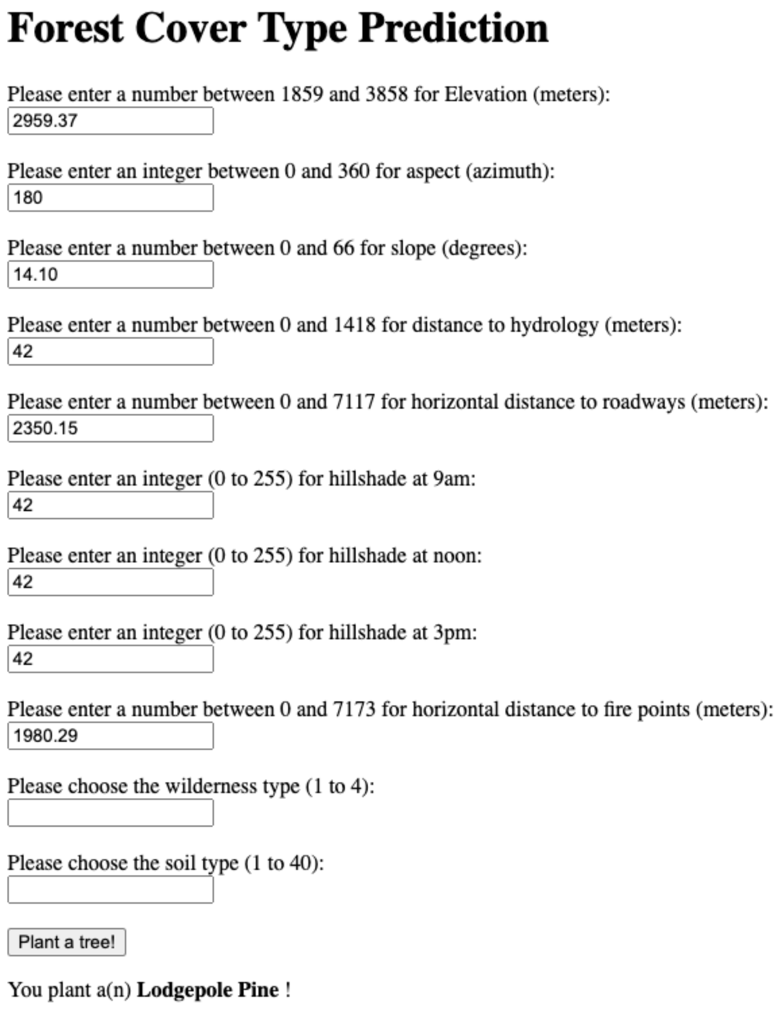
1.3.3 Other Models

I used GridSearchCV for SGDClassifier with alpha = [0.0001,0.001], epsilon = [0.01, 0.1], and learning_rate = [‘constant’,‘optimal’]. None of the testing accuracy was greater than 70.00%. I tried SVC with kernel = “rbf”, but I did not get a result from this model. It is possible that the training set is too large (116,203) and SVC is not suitable for large datasets.

2 Final demonstration proposal

I have built a simple webapp using part of the codes from MAIS webapp workshop (https://github.com/Yuyan-C/MAIS202webapp). I plan to add pictures for each type and a map of the forest.

Figure 2.1: webapp



Forest Cover Type Prediction

Please enter a number between 1859 and 3858 for Elevation (meters):
2959.37

Please enter an integer between 0 and 360 for aspect (azimuth):
180

Please enter a number between 0 and 66 for slope (degrees):
14.10

Please enter a number between 0 and 1418 for distance to hydrology (meters):
42

Please enter a number between 0 and 7117 for horizontal distance to roadways (meters):
2350.15

Please enter an integer (0 to 255) for hillshade at 9am:
42

Please enter an integer (0 to 255) for hillshade at noon:
42

Please enter an integer (0 to 255) for hillshade at 3pm:
42

Please enter a number between 0 and 7173 for horizontal distance to fire points (meters):
1980.29

Please choose the wilderness type (1 to 4):

Please choose the soil type (1 to 40):

You plant a(n) **Lodgepole Pine** !