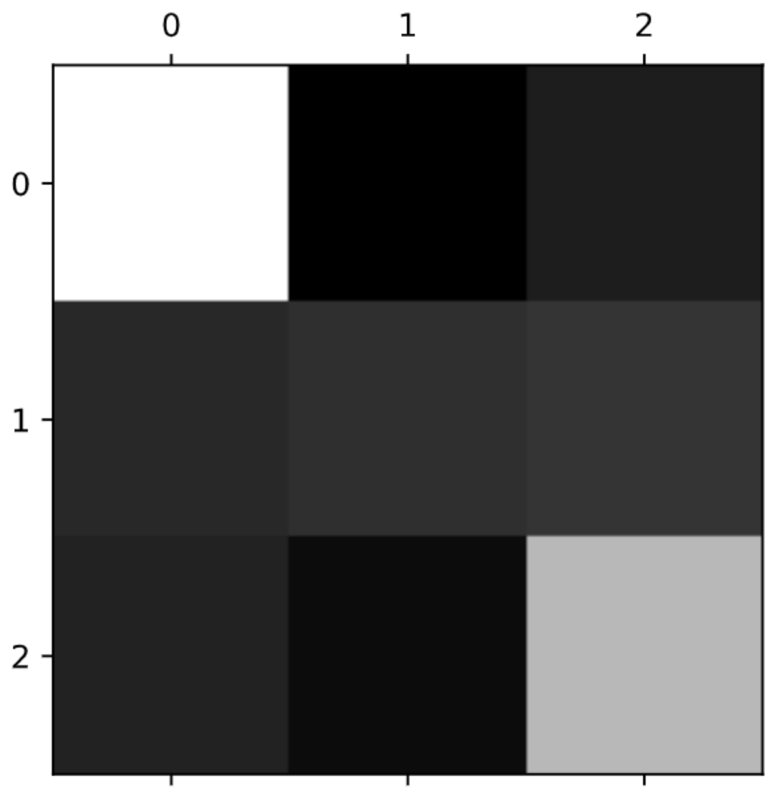
Building a multi-class classification model for imbalance data to predict burn severity using Python



Eberswalde University for Sustainable Development &
Warsaw University of Life Sciences
Master program – Forest Information Technology (FIT)
Winter semester 2020,21

Course: Programming-III
Semester: 3rd Semester
Submitted to: Prof. Dr. Luis Miranda
Submitted by: Gulam Mohiuddin, Mir Mazedur Rahaman

Contents

1. Introduction

Forest fire has emerged as one of the biggest environmental challenges globally that affect the foreset ecology, biodiversity and human life (Cortez and Morais, 2007). World is losing a Football pitch size of primary rainforest in every 6 seconds(World Resources Institute 2020). There were 58,250 wildfires occurred in 2020 that burned 10.3 million acres land. Even though the number of forest fire is showing a decreasing trend globally but the damage due to it is has a steady increasing trend in terms of forest area it damaged (Katie Hoover and Laura A. Hanson, 2021)

Scientists all over the world have been studying the forest fire trying to understand and formulating predictive analysis method for upcoming forestfires. There are different methods and approach in studying forestfire and predicting it; such as, an interval tree approach(Alberg 2015), through Spatio-temporal data mining (Cheng and Wang, 2008), using cellular automata (Karafyllidis and Thanailakis, 1997), using a Logit model (Garcia et al. 1995) etc.

Python is one of the most popular languages used in scientific computing. It is also widely used in machine learning (Varoquaux et al., 2015). Python is also popular for its use in statistics (Vallat, 2018) and predictive analysis (Bowles, 2015). In addition, python is integrated to do geospatial analysis including remote sensing (Lawhead, 2019). The dataset contains data from LiDAR and Landsat 8 raster. The data contains 37 columns and 106907 rows. Some of the advantages in deep learning model are reduction in the need for feature production and it is deemed to be one of the most time-reducing parts of machine learning approach (Gnanasekar, 2016).Traditionally, the extracted features are provided to the classification process. But, in the proposed method, it is not directly given to the classification process since it consumes more computational time to execute. Hence, the optimal feature selection module is preferred where most of the significant features are selected using Opposition-based Crow Search optimization algorithm.

On this backdrop, this project aims to build a multi-class classification model for imbalance data to predict burn severity. The specific objectives are:

- Finding out the importance feature before classification to reduce time and improve classification accuracy.
- Make a multiclass classification Model for imbalanced data to predict Burn Severity from LiDAR Metrics and asses the accuracy of the model.
- predict a dummy data to test the model accuracy.

The whole project is built on a Jupyter notebook and the code is uploaded in the GitHub (https://github.com/mazedm80/Deep-forest-Py)

2. About the data

The data that is used in this project is from the mountain forest of Oregon State in USA which is highly susceptible to forest fires because of dry forest and there was a big forest fire in 2020 that burned about 1.07 million acres and cost 354 million USD (Urness, 2020).

The data is generated from Landsat-8 image and LiDAR metrics. From Landsat-8 image the differenced Normalized Burn Ratio (dNBR) were calculated from a pre fire and post fire Landsat-8 raster and the 89 LiDAR tiles covers 30% of the total area of the dNBR area. 36 Standard metrics were created and then a excel data set were created from both dNBR and LiDAR metrics based on the overlapping area of both rasters. The LiDAR metrics were used in this are 'zmax', 'zmean', 'zsd', 'zskew', 'zkurt', 'zentropy', 'pzabovezmean', 'pzabove2', 'zq5', 'zq10', 'zq15', 'zq20', 'zq25', 'zq30', 'zq35', 'zq40', 'zq45', 'zq50', 'zq55', 'zq60', 'zq65', 'zq70', 'zq75', 'zq80', 'zq85', 'zq90', 'zq95', 'zpcum1', 'zpcum2', 'zpcum3', 'zpcum4', 'zpcum5', 'zpcum6', 'zpcum7', 'zpcum8', 'zpcum9'.

All this metrics are used as dependent variables while the burn severity used as class variable. There are three classes in the burn severity which are:

- Class 1: Unburned Area
- Class 2: Low Burned Severity
- Class 3: High Burn Severity

3. Data preparation

Firstly, the excel data is loaded into python as a data-frame. Then three columns ‘ID’, ‘x’, ‘y’ is removed because this data is unnecessary to this analysis. Then the NA values of Burn Severity which is labeled as 0 is also removed. After cleaning the data, we have 37 columns and 106908 rows of data. The data distribution of each class is as follows:

- Class=3.0, Count=36903, Percentage=34.518%
- Class=2.0, Count=28493, Percentage=26.652%
- Class=1.0, Count=41512, Percentage=38.830%

The data-frame is divided into X and y as X refers as feature variable and y as Class variable. X contains 36 LiDAR metrics and y contains the Burn Severity classes.

4. Feature Selections

Feature selection methods are intended to reduce the number of input variables to those that are believed to be most useful to a model in order to predict the target variable.Feature selection is primarily focused on removing non-informative or redundant predictors from the model (Johnson, 2013). Additionally, the performance of some models can degrade when including input variables that are not relevant to the target variable.

The most common techniques for numeric input and categorical output are correlation based, although in this case, they must take the categorical target into account.Feature selection is performed using ANOVA F measure via the *f_classif()* function.

An F-statistic, or F-test, is a class of statistical tests that calculate the ratio between variances values, such as the variance from two different samples or the explained and unexplained variance by a statistical test, like ANOVA. The ANOVA method is a type of F-statistic referred to here as an ANOVA f-test.

Importantly, ANOVA is used when one variable is numeric and one is categorical, such as numerical input variables and a classification target variable in a classification task.

The scikit-learn machine library provides an implementation of the ANOVA f-test in the *f_classif()* function. This function can be used in a feature selection strategy, such as selecting the top k most relevant features (largest values) via the SelectKBest class.

```
# define feature selection
fs = SelectKBest(score_func=f_classif, k='all').fit(X,y)
# what are scores for the features
for i in range(len(fs.scores_)):
    print('Feature name %s: %f' % (names[i], fs.scores_[i]))
```

The score of each feature is given below:

1. Feature name zmax: 257.175395
2. Feature name zmean: 313.587067
3. Feature name zsd: 711.126190
4. Feature name zskew: 144.315052
5. Feature name zkurt: 7.133150
6. Feature name zentropy: 363.657069
7. Feature name pzabovezmean: 2333.287295
8. Feature name pzabove2: nan
9. Feature name zq5: 324.358528
10. Feature name zq10: 318.357683
11. Feature name zq15: 315.573812
12. Feature name zq20: 315.429977
13. Feature name zq25: 316.211035
14. Feature name zq30: 316.940534
15. Feature name zq35: 317.451076
16. Feature name zq40: 318.011687
17. Feature name zq45: 318.591877

18. Feature name zq50: 318.904521
19. Feature name zq55: 318.687387
20. Feature name zq60: 317.753665
21. Feature name zq65: 316.222165
22. Feature name zq70: 314.152765
23. Feature name zq75: 311.546816
24. Feature name zq80: 308.518817
25. Feature name zq85: 304.822026
26. Feature name zq90: 300.229335
27. Feature name zq95: 293.940821
28. Feature name zpcum1: nan
29. Feature name zpcum2: nan
30. Feature name zpcum3: 5.949648
31. Feature name zpcum4: 6.990458
32. Feature name zpcum5: 14.444927
33. Feature name zpcum6: 25.140673
34. Feature name zpcum7: 22.999609
35. Feature name zpcum8: 24.469049
36. Feature name zpcum9: 26.780573

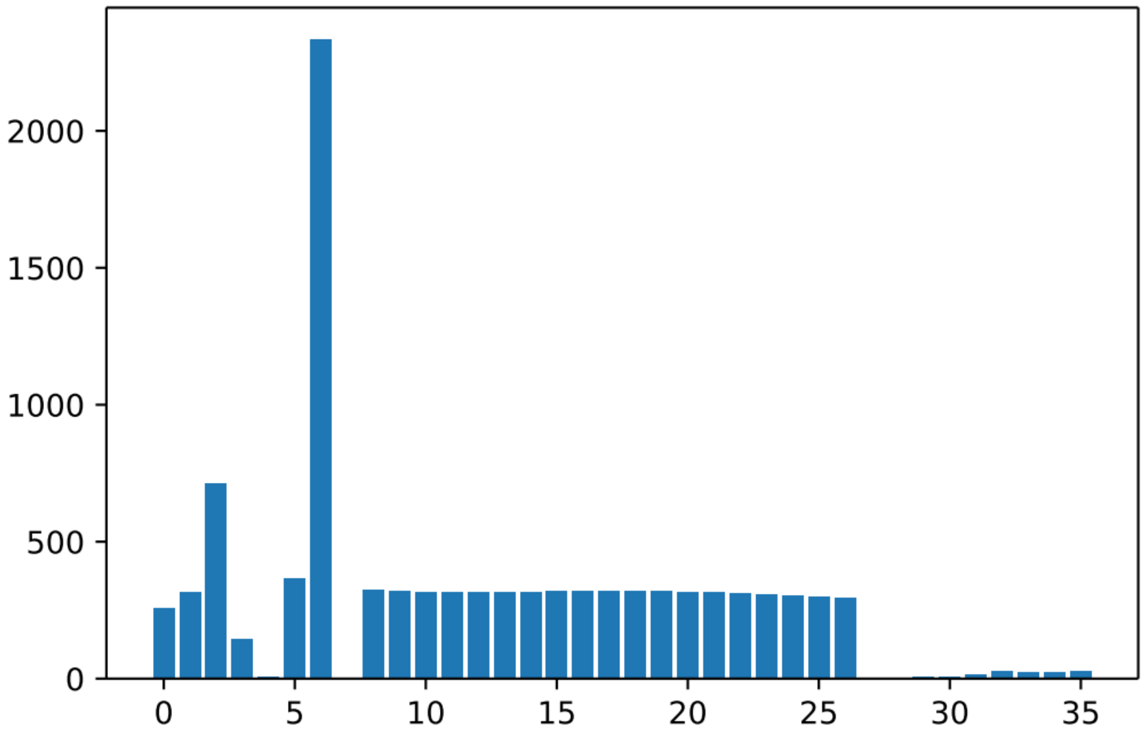


Figure 1: Bar plot of Features scores

For this case I try to manually select feature based on my observation and selects feature which score is more than 40. In this case the total number of features are 25.

Instead of guessing, we can systematically test a range of different numbers of selected features and discover which results in the best performing model. This is called a grid search, where the *k argument* to the *SelectKBest* class can be tuned.

It is good practice to evaluate model configurations on classification tasks using repeated stratified *k-fold cross-validation*. We will use three repeats of 10-fold cross-validation via the *RepeatedStratifiedKFold* class.

```
# define the evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define the pipeline to evaluate
model = LogisticRegression(solver='liblinear')
fs = SelectKBest(score_func=f_classif)
pipeline = Pipeline(steps=[('anova',fs), ('lr', model)])
# define the grid
grid = dict()
grid['anova__k'] = [i+1 for i in range(X.shape[1])]
# define the grid search
search = GridSearchCV(pipeline, grid, scoring='accuracy', n_jobs=-1, cv=cv)
# perform the search
results = search.fit(X, y)
```

5. Classification model

Random Forest is a popular and effective ensemble machine learning algorithm.It is widely used for classification and regression predictive modeling problems with structured (tabular) data sets, e.g. data as it looks in a spreadsheet or database table.

Random forest involves constructing a large number of decision trees from bootstrap samples from the training dataset, like bagging.Unlike bagging, random forest also involves selecting a subset of input features (columns or variables) at each split point in the construction of the trees.

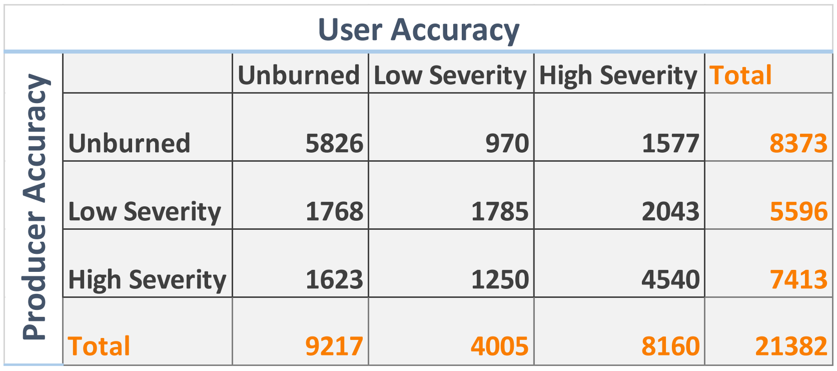
```
# define RF model to evaluate
model = RandomForestClassifier(n_estimators=1000)
# fit the model
history = model.fit(X_train, Y_train)
# evaluate the model and store results
scores = evaluate_model(X_validation, Y_validation, model)
print('>Mean Accuricy %.3f (%.3f)' % (mean(scores), std(scores)))
```

For this classification model the number of estimators is set to 1000. And for classification 70% data is used while, 30% data is used for validations.

6. Validating the model

For feature importance if we see the **F-test**scores of each feature there are very unusual difference in the scores. From normal observation it is difficult to choose the correct number of features. However, the grid search algorithm helps us to choose the best number of features for classification model. After 10 fold cross validations the result shows that **Best Mean Accuracy** is **0.489** and Best Config is selecting all **36** features.

The Random Forest classification mean accuracy is 57%. Which is pretty good considering the imbalanced data-frame. The Confusion Matrix is given below for this model.



User Accuracy

Producer Accuracy

	Unburned	Low Severity	High Severity	Total
Unburned	5826	970	1577	8373
Low Severity	1768	1785	2043	5596
High Severity	1623	1250	4540	7413
Total	9217	4005	8160	21382

Figure 2: Confusion Matrix

```
# known class "2.0" Low Burned
row = [1212.175537,1201.414185,3.744570017,0.025900438,2.527083158,0.309110
761,51.14349365,100,1196.758057,1197.177246,1197.437256,1197.896973,1198.84
1553,1199.130127,1199.453979,1199.648071,1200.200073,1200.666382,1201.11718
8,1201.655151,1202.256958,1203.067627,1203.952515,1204.922974,1205.840088,1
206.836426,1208.060669,0,0,0,0,0,0,0,0,0]
yhat = model.predict([row])
label = label_encoder.inverse_transform(yhat)[0]
```

We can use our Random Forest model to make predictions for new data by calling the *predict()* function. This will return the encoded class label for the dummy data. We can then use the label encoder to inverse transform to get the class label.

```
>Predicted=2.0 (expected 2.0)
```

As we can see the model predicted correct outcome.

7. Conclusion

In this project, we investigated the use of the feature selection approach to ensure efficient number of feature use for classification approach to the datasets with highest accuracy possible. The F-test is very useful algorithm to select the sufficient number of features. The Random Forest algorithm makes this classification more effective to predict forest fire burn severity using LiDAR metrics for unburned area. This model will be helpful to estimate and predict burn severity of any given area which is not burned yet.

8. Bibliography

1. Alberg, Dima (2015): An Interval Tree Approach to Predict Forest Fires using Meteorological Data. In *IJCA* 132 (4), pp. 17–22. DOI: 10.5120/ijca2015907398.

2. arXiv.org (2014): Image Fusion Techniques in Remote Sensing, updated on 3/24/2014, checked on 2/19/2021.

3. Bisquert, Mar; Caselles, Eduardo; Sánchez, Juan Manuel; Caselles, Vicente (2012): Application of artificial neural networks and logistic regression to the prediction of forest fire danger in Galicia using MODIS data. In *Int. J. Wildland Fire* 21 (8), p. 1025. DOI: 10.1071/WF11105.

4. Bowles, M., 2015. *Machine learning in Python: essential techniques for predictive analysis*. John Wiley & Sons.

5. Cheng, Tao; Wang, Jiaqiu (2008): Integrated Spatio-temporal Data Mining for Forest Fire Prediction. In *Transactions in GIS* 12 (5), pp. 591–611. DOI: 10.1111/j.1467-9671.2008.01117.x.

6. Chowdhury, Ehsan H.; Hassan, Quazi K. (2015): Operational perspective of remote sensing-based forest fire danger forecasting systems. In *ISPRS Journal of Photogrammetry and Remote Sensing* 104, pp. 224–236. DOI: 10.1016/j.isprsjprs.2014.03.011.

7. Cortez, P. and Morais, A.D.J.R., 2007. A data mining approach to predict forest fires using meteorological data.

8. Gnanasekar, P. A. N. (2016). Investigation on feature extraction and classification of medical images.

9. Garcia, C. V.; Woodard, P. M.; Titus, S. J.; Adamowicz, W. L.; Lee, B. S. (1995): A Logit Model for Predicting the Daily Occurrence of Human Caused Forest-Fires. In *Int. J. Wildland Fire* 5 (2), p. 101. DOI: 10.1071/WF9950101.

10. Ghassemian, Hassan (2016): A review of remote sensing image fusion methods. In *Information Fusion* 32, pp. 75–89. DOI: 10.1016/j.inffus.2016.03.003.

11. Gigović, Ljubomir; Pourghasemi, Hamid Reza; Drobnjak, Siniša; Bai, Shibiao (2019): Testing a New Ensemble Model Based on SVM and Random Forest in Forest Fire Susceptibility Assessment and Its Mapping in Serbia's Tara National Park. In *Forests* 10 (5), p. 408. DOI: 10.3390/f10050408.

12. Johnson, J. M. K. (2013). Applied Predictive Modeling.
13. Karafyllidis, I. and Thanailakis, A., 1997. A model for predicting forest fire spreading using cellular automata. *Ecological Modelling*, *99*(1), pp.87-97.
14. Katie Hoover; Laura A. Hanson: Wildfire Statistics. Available online at https://fas.org/sgp/crs/misc/IF10244.pdf, checked on 2/19/2021.
15. Lawhead, J., 2019. *Learning Geospatial Analysis with Python: Understand GIS fundamentals and perform remote sensing data analysis using Python 3.7*. Packt Publishing Ltd.
16. Sherstjuk, Vladimir; Zharikova, Maryna; Sokol, Igor (2018 - 2018): Forest Fire-Fighting Monitoring System Based on UAV Team and Remote Sensing. In : 2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO). 2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO). Kiev, 4/24/2018 - 4/26/2018: IEEE, pp. 663–668.
17. Sunar, F.; Özkan, C. (2001): Forest fire analysis with remote sensing data. In *Int. J. Remote Sensing* 22 (12), pp. 2265–2277. DOI: 10.1080/014311601300229818.
18. Urness, Z., 2020. *Oregon's 2020 wildfire season brought a new level of destruction. It could be just the beginning*. [online] Eu.statesmanjournal.com. Available at: <https://eu.statesmanjournal.com/story/news/2020/10/30/climate-change-oregon-wildfires-2020/6056170002/> [Accessed 4 February 2021].
19. Vallat, R., 2018. Pingouin: statistics in Python. *Journal of Open Source Software*, *3*(31), p.1026.
20. Varoquaux, G., Buitinck, L., Louppe, G., Grisel, O., Pedregosa, F. and Mueller, A., 2015. Scikit-learn. *GetMobile: Mobile Computing and Communications*, 19(1), pp.29-33.
21. Wood, David A. (2021): Prediction and data mining of burned areas of forest fires: Optimized data matching and mining algorithm provides valuable insight. In *Artificial Intelligence in Agriculture* 5, pp. 24–42. DOI: 10.1016/j.aiia.2021.01.004.
22. World Resources Institute (2020): We Lost a Football Pitch of Primary Rainforest Every 6 Seconds in 2019. Available online at https://www.wri.org/blog/2020/06/global-tree-cover-loss-data-2019, updated on 6/2/2020, checked on 2/19/2021.