Vipin Gunda
Dr. Aleksandr Michuda
Cornell University, August 2022

Regional Origin Classification using Surnames across Administrative Units

With my first summer of the Laidlaw Research and Leadership Scholars Program complete, I have the chance to reflect on my research experiences as a Laidlaw scholar at Cornell University. My project, "Regional Origin Classification using Surnames across different Administrative Units," was done under the supervision of Dr. Aleksandr Michuda, an Assistant Research Professor at the Center for Data Science for Enterprise and Society. Dr. Michuda studies how machine learning (ML) and big data can be used to solve problems in development economics, which lies at the foundation of our research on SafeBoda.

To start, it is important to recognize the rise of smartphones in developing countries over the past few years. The growing availability and ownership of smartphones in these areas directly ties into the growth of internet technologies like ride-share applications. This brings us to SafeBoda, a ride-share platform based in Uganda. SafeBoda has extensive experience in the boda-boda market in Uganda and has the proper technology necessary for collecting and maintaining drivers' data. Dr. Michuda's work aims to measure the explicit differences between SafeBoda's on- and off-platform services, which would allow us to study how the introduction of ride-share applications has transformed the Ugandan labor market. He also explores the links between the rural and urban sectors, which has great potential to affect policy design around these platforms. To reflect my interest in Computer Science and accommodate for Laidlaw's six-week research timeframe, Dr. Michuda and I designed a more abbreviated, programming-focused project.

My project was centered around using machine learning and drivers' name data to predict SAP regional origin at different administrative units: SAP SAP region[1], GAUL[2], and district[3]. This research is driven by performance changes for different numbers of administrative units, which could possibly motivate the switch to treat this as a regression problem instead of classification for improved performance. My project highlights one part of the tradeoff we are after in the long-term: looking at how more administrative units affect weather data precision. Taken from Dr. Michuda's larger study, a compilation of drivers' surname, SAP region, GAUL, and district, along with other demographic and geographic information, was sent to me as the primary dataset for my analysis. The dataset was in the form of a feather file, a convenient way to store dataframes, meaning that you can generally work with it as you would a typical comma-separated values file (CSV). The dataframe had 14,589,193 rows, so we had demographic and geographic data for over fourteen million drivers.

The first step in the process was setting up a remote environment to run code. Due to the limitations of personal computers, working with such large datasets is typically done on computing clusters, which have the necessary computing power and storage for such tasks. With the help of Dr. Michuda, I learned how to use Secure Shell (SSH), a means of network communication between a computer and a remote host. On Visual Studio Code, an Integrated Development Environment (IDE) that makes it easy to compile, run and debug code, we set up an SSH connection to whale.orie.cornell.edu, or whale for short. Using whale, users' code runs significantly faster and thwarts storage caps and related errors as frequently.

---

[1] SAP Regions are administrative units that break up Uganda into 4 parts: North, West, Central and Eastern.
[2] GAUL units (Global Administrative Unit Layers) break up Uganda into 10 parts.
[3] Districts break Uganda up into 112 parts.

The next step toward analysis was processing the data. Oftentimes, driver surnames had noise, with the primary culprit being miscellaneous symbols. However, this is expected due to natural errors from user input. Using a function that replaces spaces, hyphens, apostrophes, periods, and zeros (which users sometimes mistake for O's), the surname column of the dataframe was processed. Processing all 14,589,193 rows in the dataframe took 58.2 seconds. The SAP Region, GAUL, and district data were not processed because it was much more structured.

```
df = (
    df_aux.loc[lambda df: df[features]
            .str.replace('-','')
            .str.replace(' ','')
            .str.replace("'",'')
            .str.replace('.','')
            .str.replace('0', 'O')
            .str.isalpha()
            ]
    .loc[lambda df: df[features].apply(lambda x: len(x)>=2)]
    )

return df
```

Figure 1: Code Snippet from the Data Processor

With the data processed, we set up the primary machine learning model. The model that I used came from XGBoost, a Python library that stands for Extreme Gradient Boosting. Gradient boosting relies on the process of choosing the best potential model, such that when it's combined with previous models, there is a minimization in overall prediction error. Specifically within XGBoost, I used the XGBClassifier, which is one of the leading ML models for classification. Classification is a type of machine learning problem where a label is predicted for a given instance of input data. In my case, the input data is the surnames, which are also referred to as the features for classification. Meanwhile, the class that the model is trying to predict based on the input is the SAP region, GAUL, or district, often referred to as the labels for classification.

From here, we can set up some other tools that are important for analysis. The first is a TF-IDF Vectorizer, which comes from Scikit-Learn, a machine learning library that contains useful tools for data processing and classification. TF-IDF stands for term frequency-inverse document frequency, which indicates that the vectorizer compares the number of times a word appears in a document to how many documents that word appears in. This results in a way to measure the importance, or weight, of each of our features (i.e. the importance of a certain driver's surname). A vectorizer is a tool that converts text data to vectors, allowing it to be used and understood by machine learning. In our case, a TF-IDF Vectorizer is more ideal than a Count Vectorizer. While a Count Vectorizer transforms a given text into a vector based on the frequency of each word, the TF-IDF vectorizer is also able to condition on importance, with a reference to the math behind TF-IDF seen in figure 2.

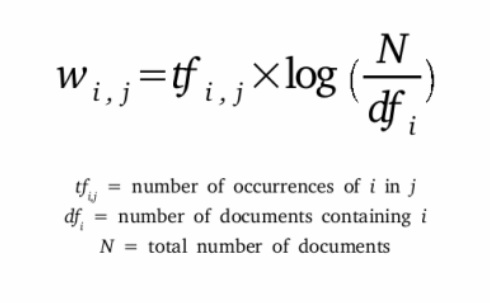

Figure 2: TF-IDF Weight Calculation

Another tool that we configured is the StratifiedGroupKFold, which also comes from the Scikit-Learn library. A KFold is a way to split data into k sections, which is useful for splitting our data for training, validation, and testing. A StratifiedGroupKFold, specifically, creates k folds that preserve the percentage of samples for each class and also ensures that the same group will not appear in multiple folds. With the StratifiedGroupKFold set up, we can use the TF-IDF vectorizer on the surnames. Say we call the transformed surnames, X. We can assign unique labels (4 SAP regions, 10 GAULs, and 112 districts) to the variable y each time. In ML, X and y

are conventional names to represent the features (independent variable) and targets (dependent variable, i.e. labels) respectively. Running X and y into StratifiedGroupKFold creates a splitter that will be useful in the next step of the process: creating a calibrated model.

To create a calibrated model, we use CalibratedClassifierCV, also from the Scikit-Learn library. CalibratedClassifierCV takes in our XGBClassifier, splitter, along with other parameters. It produces a calibrated model which supports probability prediction. Using the generated model and XGBoost's plot_importance feature, we can create a Feature Importance chart.

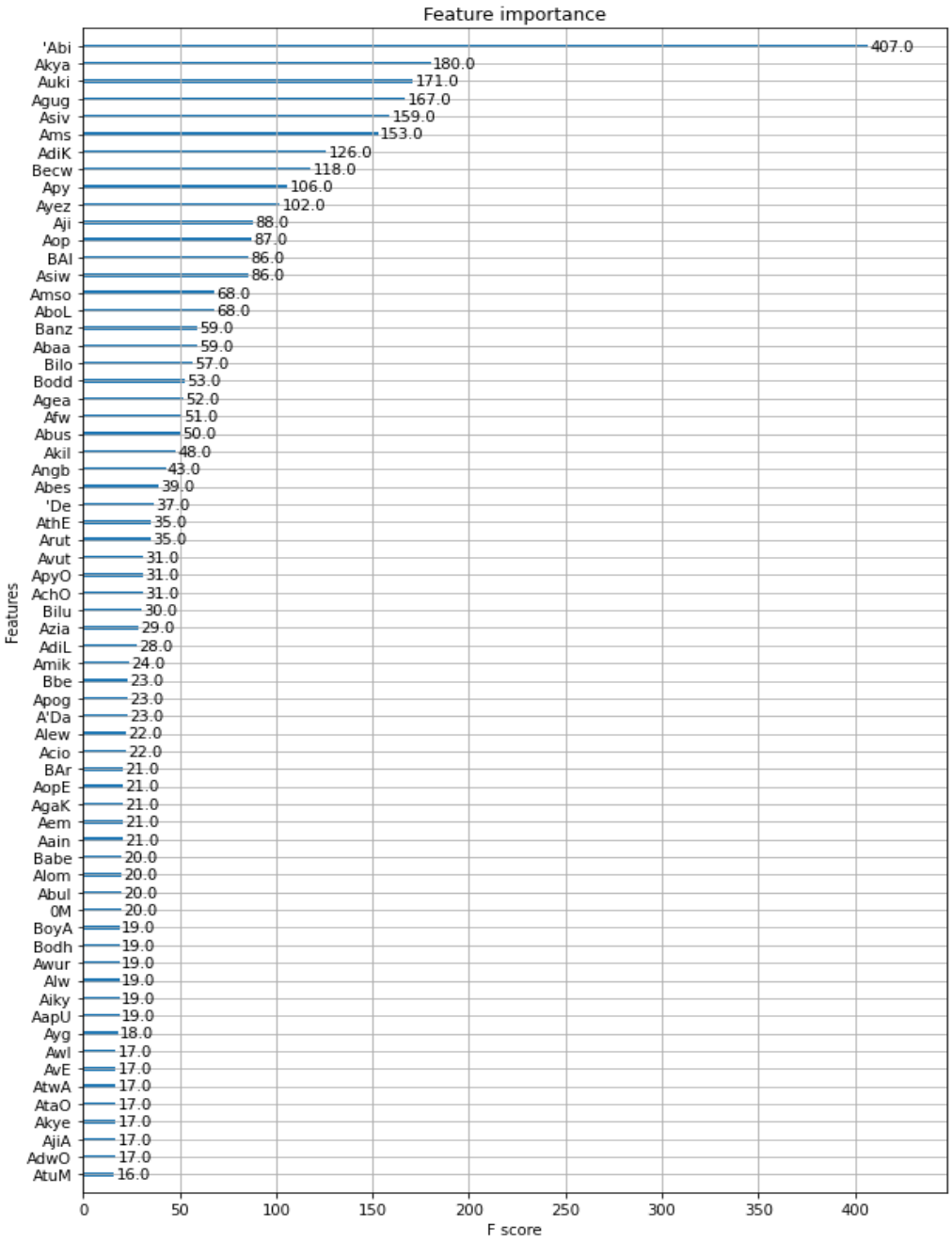

Figure 3: Feature Importance Chart for Region

Feature importance refers to the measure of how useful a certain input feature is to predict the target variable. The F score measures the increase in a model's prediction error after permuting the feature. For our model's Feature Importance Chart (Figure 3), the target is the SAP region while each surname is a feature. From the graph, we find that names that contain "Abi" have the greatest impact on which SAP region the model predicts. Since "Abi" has the highest F score, it is a much more powerful feature in the sense that out of all features, the model would do the best with it and the worst without it. Another interesting aspect of the chart to consider is how the feature with the second-highest F score has a score that is less than half of the highest F score. Despite this large difference between "Abi" and "Akya," the differences between the later features are much less drastic. Looking at the graph, we can see that the lower half of features in the figure have very similar F scores with little difference between each pair. And so, Feature Importance Charts like Figure 3 can provide insights into which features are the most useful along with a unique visual representation that has the potential to reveal trends or patterns in feature importances.

The next step in our analysis is to generate learning curves. Using the calibrated model from earlier, we can pass it into the LearningCurve function from Scikit-Learn and YellowBrick, an extension of Scikit-Learn that generates visualizations. A learning curve is useful because it depicts the relationship between the training score versus the cross-validation score across a range of training samples. From this graph, we can gauge the benefits of having more data since we can find the point at which there is enough data such that there is no benefit from having any more data. And so, I generated learning curves for SAP region, GAUL, and district prediction.

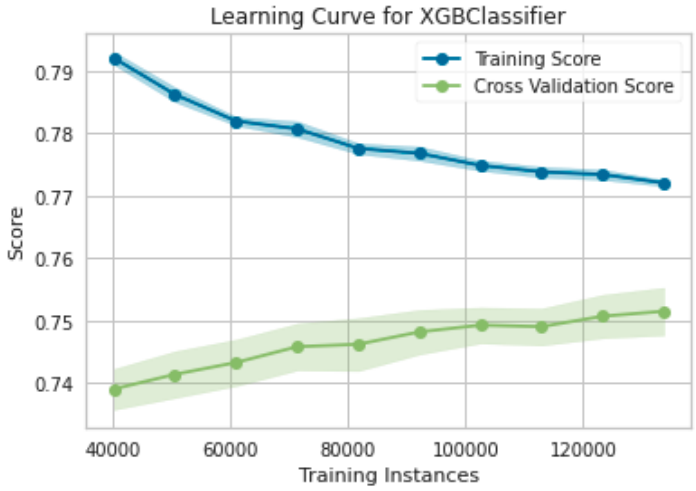

Figure 4: Region Learning Curve

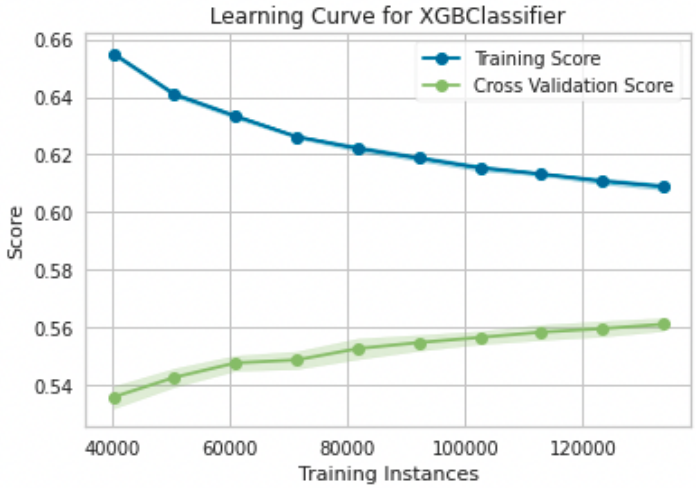

Figure 5: GAUL Learning Curve

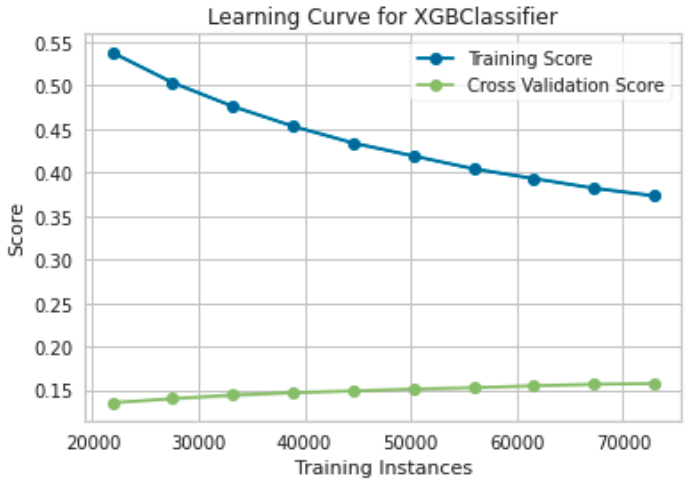

Figure 6: District Learning Curve

Note that the learning curves were run on one-hundredth of the data. Even with whale's incredibly powerful computers, running these learning curves took three to four hours each, with the district learning curve taking even longer than that. After trying to run an SAP region learning curve on the entire dataset for a day, I was met with a storage error so I decided to use smaller datasets to extrapolate from.

Between Figures 4, 5, and 6, we can see some similarities. For instance, training scores always seem to be higher than cross-validation scores. And by design, both lines in every graph narrow in, indicative of convergence. However, we can also see how the scores differ based on the chosen target. The SAP region accuracy seems to be converging towards 0.76, while GAUL is heading towards 0.59, and finally, district merging at around 0.27. This suggests that the accuracy of the final optimal model from greatest to least will be SAP region, GAUL, and finally district. The reason behind this trend lies in the number of labels for each geographic denomination. Four SAP regions are much easier for the model to train and learn on rather than 112 districts. Furthermore, we might expect names to vary more between SAP region to SAP region rather than between neighboring districts.

Another point of analysis is finding the number of instances where the lines converge, indicating that we would not benefit from any more data. For the SAP region learning curve, the two lines are just 0.03 away, while the difference is around 0.05 for GAUL and 0.2 for district. It is clear that they all need more instances for us to pinpoint the exact number where they converge. However, we can predict that the district learning curve will need many more samples to converge than SAP region or GAUL.

Lastly, we generated confusion matrices. Confusion matrices can be generated through Scikit-Learn and Yellowbrick just like learning curves can. Using our calibrated classifier along

with a set of test X and y values, a confusion matrix shows how each of the test values predicted classes in comparison to their actual classes. This type of visualization is useful because it allows us to see which classes are easily confused. The confusion matrices were run for one-hundredth of the data once again.

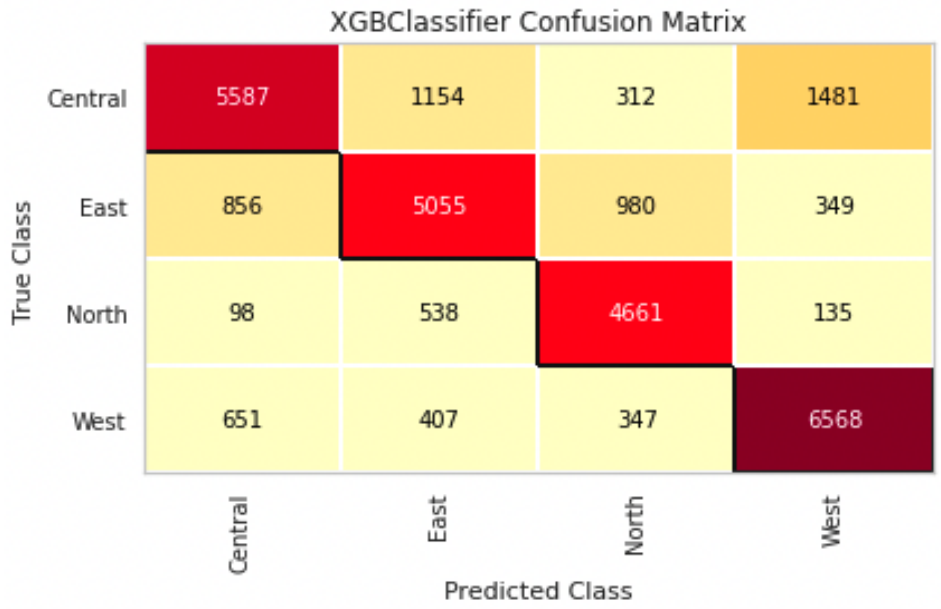

Figure 7: Region Matrix

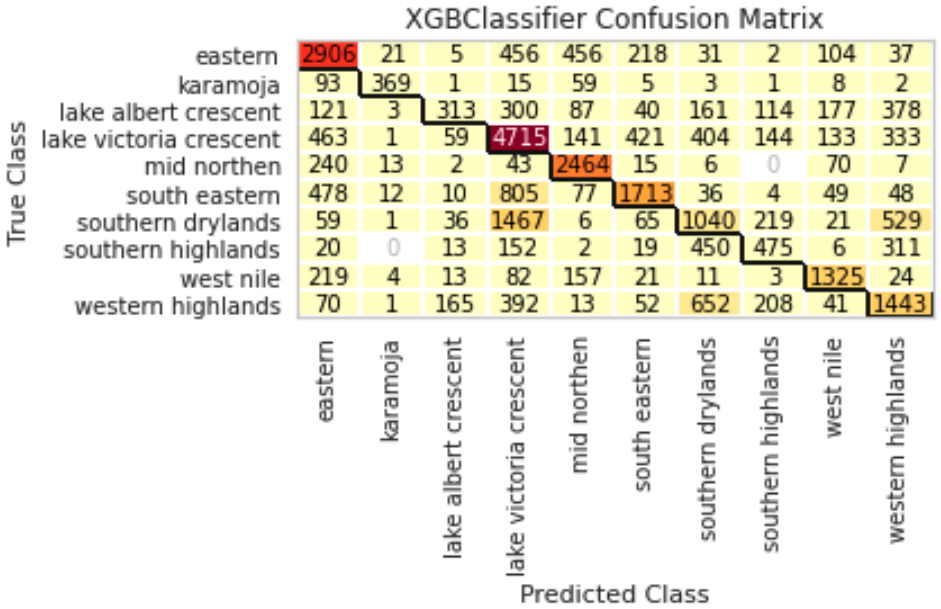

Figure 8: GAUL Matrix

The diagonals of figures 7 and 8 indicate the instances where the model correctly predicted the class. All boxes outside of the diagonal are incorrect predictions. Taking the color map into consideration, we see that the diagonal of figure 7 is clearly emphasized, indicating that the model did well in predicting the true class. Looking for the brightest color outside of the diagonal in figure 7, we notice the box on the top-right. So for predicted "West" 6,568 predictions were right, but 1,481 were actually "Central," which results in the largest misclassified box on the figure. As we move to figure 8, we see another interesting point of analysis: the diagonal is not as emphasized in the color map as in figure 7. This indicates that in general, our model was worse at choosing the GAUL than the SAP region, which is a relationship we saw previously in the scores from the learning curves. We also see that the GAUL model did the best with predicting "Lake Victoria Crescent." Another interesting aspect of figure 7 is that predicted "Karamoja" was never truly "southern highlands" and predicted "southern highlands" was never truly for "mid-northern" Through confusion matrices like these, we can pinpoint which classes our model needs to improve on and perhaps even discover how our model interprets Uganda by analyzing trends that we see.

With the analysis complete, the next step going into the future will be to turn this into a regression problem, meaning that instead of predicting classes that represent geographic denominations, we will be working directly with GPS coordinates. From there, the goal will be to connect weather data for those coordinates and ideally, produce GPS predictions for the drivers. The societal implications behind this work are that we can capture the connection between the rural and urban sectors of the economy more precisely. This is important for designing policies to help workers in cities that need to provide remittances to their family members in rural areas, which is crucial when rural households have no means of protecting

themselves against shocks such as drought. And so, I will look to expand the scope of the project now that the preliminary analysis is complete.