1 Introduction [15 points]

- Group Members: Thai Khong and Sasha Fomina
- Team Name: disco_panda
- Private Leader Board Place: 13th out of 33
- AUC Score: 0.82963
- Division of Labor:
  - Sasha briefly explored Logistic Regression and Random Forest, but focused on the Adaboost and LightGBM models. Sasha also performed some exploratory data analysis, implemented data transformations of the geospatial features, and experimented with under- and oversampling.
  - Thai initially performed some exploratory data analysis to decide on what features to use. Thai also performed some feature engineering, including manipulating and categorizing temporal and geospatial data. Thai then focused on training XGBoost models using various techniques: tuning hyperparameters, exploring techniques that handled imbalanced datasets, and ensemble learning.

2 Overview [15 points]

In the initial phase of our project, we performed data exploration which allowed us to notice temporally periodic trends, geospatial trends, and an imbalanced class distribution. These observations together with feature importance function of our models and correlation heat maps of features motivated our approached to feature manipulations and our final features were as follows:

- LightGBM: Latitude, Longitude, Discovery_Time, FIPS_Code, Fire_Size, Polar_R, Polar_theta, Prev_-Fire_Size, PCA_Y
- XGBBoost: Same without FIPS_CODE and with FIRE_Class

We also used simple data imputation, filling in NaNs that were mostly due to Discovery Time with the mean, after noticing that dropping null valued data would significantly reduce the number of training examples. While we experimented with undersampling and oversampling techniques using the RandomUnderSampler and SMOTE, respectively, we found that our cross validation and test scores were not improved by this technique

We then tried various models, some briefly such as Logistic Regression and Random Forest models, and some extensively like AdaBoost, LightGBM and XGBoost. We eventually settled with optimizing LightGBM and XGBoost models, as they performed the best out of all models. We performed an exhaustive grid search (on XGBoost) and a random search (on LightGBM) to tune the hyperparameters, checking the cross-validation score for each set of parameters.

Regarding the work timeline, we performed the initial data exploration leading to the creation of the Year and Month features after noticing seasonal trends in the data. After briefly exploring AdaBoost and RandomForests, we decided to focus on XGBoost and LightGBM. We used grid/random cross validation search for XGBoost and LightGBM. When the models' performances began to plateau regardless of the hyperparameters, we explored techniques that accounted for the imbalanced nature of the dataset such as under- and over-sampling, a combination of both, or assigning class weights based on the ratios of labels. When none of these methods improved the performance of our model, we tried ensemble learning, particularly stacking models and observed no noticeable performance. Finally, including more feature engineering, namely adding the polar coordinates, PCA coordinates, and previous fire size, improved our performance.

3 Approach [15 points]

An important initial step in our data manipulation was using the sklearn SimpleImputer and imputing with the mean value over the missing feature, in order to avoid removing more than half of the training data. Additionally, the monthly and yearly periodic trends in the fraction of different wildfire causes (see Figure # in Colab) motivated us to avoid treating each date-time point as joint year-month-day feature. Instead, we separated the date-time data for each training example into a Year and a Month feature, creating a seasonal correspondence between data points in the same month, but different years. We also scaled down the year feature to denote years after 1991 to emphasize the sequential relationship between years rather than their magnitudes. These temporal categorization features were significant features (high feature importances) and scaling also improved performance. Moreover, in an attempt to factor in a potential relationship between fires in a given year and fires in the year prior, we added a feature called Prev_Fire_Size, which contained the mean of the previous year's fire sizes.

An improvement in performance was generated by adding polar coordinate features in addition to the Cartesian latitude and longitude features. This decision was driven by the fact of the strong geospatial relationship of the causes of fires, as was evident by the high feature importances of latitude and longitude in XGBoost and LightGBM. With polar features, we allowed our tree-based methods more axes to separate along spatial trends in the data. Adding the y-coordinate of the sklearn PCA transformed Latitude and Longitude, was similarly motivated and improved performance by, again, giving the model an additional rotational coordinate that was not too strongly correlated with the other spatial features.

Regarding the choice of models, we focused on tree-based boosted models. This choice was driven by the interpretability of the tree-based models, which together with the feature_importance_ functions of these models helped us make sense of next steps for improvement. Moreover, the boosted ensemble methods give the added benefit of preventing overfitting with bagging parameters and relatively non-complex base estimators, while improving accuracy with more estimators. After experimenting briefly with the sklearn RandomForestClassifier and the sklearn AdaBoostClassifier, we found that XGBoost and LightGBM had considerably faster learning speeds and performed better in cross validation and test. Since both XGBoost and LightGBM have many more parameters than the classic gradient boosting, the process of training involved a lot of hyperparameter tuning through approaches like GridSearchCV (sklearn) and custom-implemented random search CV. This was a disadvantage of the model because a completely exhaustive search of hyperparameters is impossible. Another disadvantage is that regularization must be done carefully in the form controlling the tree depth and using bagging in order to prevent overfitting.

As the cross validation and test error of our models started plateauing, we began analyzing the performance of our models, particularly by using the plot_confusion_matrix function from sklearn.metrics to detect true/false positives and negatives. We noticed that all models had false negatives on wildfires labeled 3 and 4. In particular, the Adaboost model did not predict any 3's in the data at all (see Colab for details). Through this, we discovered the imbalanced class distribution in the data with far more 2-labels. We tried to use undersampling and oversampling via RandomUnderSampler and SMOTE from imblearn as well as stacking models. However, this led to fewer false positive 2's but more false negative 2's, and we decided to prioritize predicting 2's correctly since the test distribution would likely also be biased toward 2's.

4 Model Selection [15 points]

As the LightGBM and XGBoost models trained, we set them to optimize the softmax function as their objective function, as this function is a logistic function that can be generalized to multiple dimensions, making it suitable for the purpose of multiclass classification. Note that we split our training data using a stratified 5-fold cross-validation scheme (StratifiedKFold from sklearn.model_selection) and evaluated our models with the cross_val_score and cross_validate functions from the sklearn.model_selection library using the area under curve (AUC) of the receiver operating curve (ROC) metric. We chose stratified 5-fold cross validation, because we wanted to maintain the same ratio of classes across each CV split since the class distribution in the training set was very imbalanced. In this manner, our stratified 5-fold cross validation scores acted as good proxies for the test score when we would compare models with different hyperparameters to each other during grid-search or random search. We also made sure to set "return_-train_score" to True in our cross validation, so that we could compare the training error and validation error to check for overfitting and tune regularization hyperparameters accordingly. Cross Validation drove the scoring of our models and the tuning of hyperparmeters, because it was impossible to be exhaustive in our search so we had to tune the ranges of the grid-search and random search. Some important parameters were the number of leaves or the maximum depth of each tree estimator, which helped improve AUC when paired with proper regularization like bagging fraction and frequency to avoid overfitting.

Also, we chose to use the AUC ROC metric with the one-vs-rest option as this was used to score our models on the Kaggle leaderboard. We also would use plot_confusion_matrix (sklearn.metrics) in tandem with cross validation to gauge our models' performances. The confusion matrix was a useful tool in understanding the ROC AUC score, since we can see the true positive and false positive rates for each label. Overall, our evaluation procedure involved finding the model's Stratified-5-fold cross validation score, plotting its confusion matrix and, if we were happy with these results, retraining the model on the full training set before submitting a prediction of the test set onto Kaggle to be scored.

After evaluating initial baseline models, conducting an exhaustive grid-search or random search and doing some additional feature engineering to optimize the performance of our models, the following LightGBM and XGBoost models performed best (compared respectively to LightGBM and XGBoost models with different hyperparameters), at the corresponding evaluation criteria:

Mean Stratified 5-Fold Cross Validation

- LightGBM: num_leaves:280, bagging_fraction:0.5, bagging_frequency:3, feature_fraction:0.9, learning_rate: 0.1, lambda_l2:0.510204, objective: 'multiclass', num_class:4 **(score:0.8376)**

- XGBoost: objective: 'multi:softprob', max_depth:12, n_estimators:500, gamma:5, min_child_weight:7, subsample:0.7, colsample_bytree:0.7, and learning rate:0.05 **(score:0.842)**

5 Colab and Piazza link [15 points]

Colab link: disco_panda.ipynb
Piazza link: Public Piazza link

6 Conclusion [15 points]

To summarize, tree-based boosted models like LightGBM and XGBoost yielded the highest performance at categorizing wildfires using the AUC-ROC evaluation metric. Based on the results of our models, the top 10 features that influenced the test predictions of our model were, ranked in decreasing importance order: Longitude > Polar_R > Polar_Theta > Latitude > PCA_Y > Year > Discovery_Time > Month > Fire_Size > Prev_Fire_Size. We noticed that Longitude and Latitude had the largest importance since they were main factors that formed separating boundaries between clusters of fires located in different states and clusters of different fire labels within specific regions (See Colab demo). Since the polar coordinates and PCA y-coordinates were derived from these features, they also contributed a significant amount to the performance of the model by allowing the decision tree to split data points using different axes rather than just the two formed by latitude and longitude. Date-time factors such as Year, Discovery_Time and Month also had noticeable influence due to the seasonal nature of the wildfires as discussed in the previous sections. Fire_Size and Prev_Fire_Size had some influence, but to a lesser degree since the wildfire sizes across labels varied wildly.

Some of the lessons we learned from this project were:

1. to balance feature engineering vs. model selection and tuning. It is important to start out with a good model, but as our models' performances stagnated despite tuning hyperparameters and trying out different techniques, the only thing we did that improved the models' performance was adding new features.
2. to extract additional useful information from existing features, especially those exhibiting high importance features such as date-time information like day of week, year and month, or geospatial information like polar coordinates, PCA y-coordinates. This allowed us to capitalize on their usefulness by creating extra features that contributed to the overall performance of the models.
3. to recognize and handle imbalanced data sets using under- and over-sampling techniques and using ensemble learning

Some of the challenges we faced were:

1. struggling with performance plateaus and not taking the correct course of action to address it. We should have recognized that, as we continued to tune our hyperparameters and fail to improve our models, that the issue was most likely in the lack of features rather than the optimality of our models.
2. spending too much time on oversampling and undersampling instead of feature engineering. Similar to point (1), rather than focusing solely on such techniques to address the imbalanced distribution of the wildfire labels, we should have done more feature engineering instead.
3. underutilizing ensemble learning. We were aware of the spatial separation of the data, but did not recognize that it would have been a great idea to train two models separately on the two different states (CA and GA) or even two models with an imbalanced dataset and a resampled one and combining them using blending to improve our predictions and address some of our aforementioned issues.

7 Extra Credit [10 points]

The AUC metric is an effective metric for muti-class classification problems, because it evaluates how capable the model is of distinguishing between classes. This metric is able to do integrating over the ratio of the true positive rate to the false positive rate at different thresholds for each of the classes. In this fashion, it is an appropriate metric to use for this challenge as it was a multi-class classification problem. Moreover, our challenge involved an imbalanced class distribution which makes AUC a much better evaluation metric than accuracy, because an accuracy metric would favor predicting correctly on the majority class, while AUC instead favors the ability to discriminate between different classes. However, the fact that our data was incredibly imbalanced, with 3x more 2-labels than any other label data point, means that perhaps AUC will be overly optimistic. This is because the false positive rate is made smaller by the heavily imbalanced data due to a large number of true negatives. Instead, using the Average Precision metric, which is the area under the Precision-Recall curve, might avoid this issue.