Predicting Wildfires in BC

Introduction

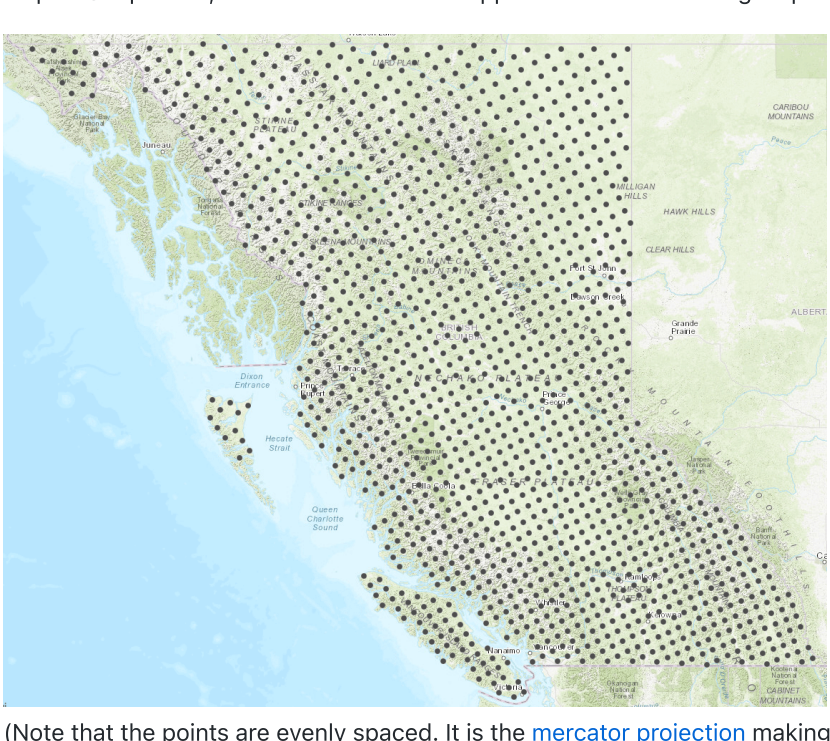
Each year, BC has numerous wildfires. They result in lost property and, tragically, often loss of life. The cost of fighting the fires is also incredibly high - most years in the hundreds of millions of dollars. In 2020, for example, the cost was $213.8 million despite a relatively quiet fire season.

What if fires could be predicted before they happen? The province could then plan better prior to each fire season, and areas likely to burn could be pre-emptively put on evacuation alert, saving lives.

Approach

For a more realistic approach, we decided that we would train our models on all data up to 2015, and then use 2015-2020 as our test data. This would allow us to simulate predicting fires prior to the fires occuring and see whether our model might be useful on real-world data. The number of fires and area burned from 2015-2020 varied significantly year to year, and our hope was that the model would work well in both scenarios.

One of the initial challenges we faced was figuring out how to split up the province into areas where we could predict fire / no fire. The approach we decided on was overlaying evenly spaced points on top of the province (each point 25km apart) with the help of Geopandas, which is Pandas with support and functions for geospatial data. The resulting points looked like this:



(Note that the points are evenly spaced. It is the mercator projection making it look like the spacing is uneven.)

All data was then attached to the closest point. Despite being points, for the purposes of our predictions we thought of each point as a 25km x 25km square with the point at the center. Our goal was to predict fire/no fire for each square in a given year.

Data

Finding, collecting and cleaning the data took the most time (by far). Each data source was put into its own folder, cleaned and then transformed into a format that could be easily joined with other data based on a `point_id` and `year` (if the data was time-dependent, such as population). We opted for the joining method (as opposed to building upon one giant dataset) because it was the most flexible: if a data source changed, we only needed to regenerate the files for that data source, rather than re-run all steps to combine all our data back together. Sources for all data is included in `README` files for each dataset.

Province Points

To generate the overlay of points on top of the province we first needed to obtain a file containing the geometry of British Columbia. Statistics Canada publishes files containing official geographic boundaries, including those for provinces and territories. From those we were able to extract the geometry for just BC and then filter out all small polygons less than 1000 sq km (e.g. small islands).

The next step was to generate a grid of evenly spaced points, which we were able to do using Geopandas. This grid was then intersected with BC geometry to only give points within BC (see screenshot above) - this left us with a total of 1,444 points. Each point was also given a `point_id`, which we could then reference in other datasets. Loading lots of geometries into memory is slow, so during our ETL process for every other dataset we opted to spatially join it with the points dataset, keep `point_id` and drop the geometry columns. This also significantly decreased file sizes.

Fire

The BC Wildfire Service provides data on historical fires in BC from 1919-2020. There is quite a bit of data associated with each fire (mostly irrelavent metadata), so the first step was to extract the data we cared about: `year`, `fire_cause`, `area` and `location`. We then matched the center of each fire with the closest `point_id` (only the closest because large fires may span multiple points). The fire data was then grouped by `point_id` and `year` and outputted as three datasets: one for all fires, one for human-caused fires, and one for weather-caused fires, each of which we could then use for our machine learning.

When exploring the fire data, we noticed something interesting: the number of fires in the data we downloaded didn't match the official BC Wildfire Summary page. It's not even close: our fire counts are roughly 20% of those given in the summary. This was initially quite concerning, however upon further analysis we discovered that the total area burned matches almost perfectly. Our guess is very small fires might have been excluded or some fires in the same area may have been merged together in the dataset we obtained. In any case, we didn't have another source for authoritative fire data and the total area was almost identical, so we were not very concerned.

Elevation

We obtained the elevation data by repeatedly calling the Canadian Digital Elevation Model API (provided by the federal government) for each of the points we generated. Elevation affects the type of trees that are able to grow (and how densely they can grow) and is also a major factor in how quickly temperatures drop overnight. Both of those have an affect on how easily a fire might be started and how it might spread.

Weather

Weather data was some of the most annoying data to collect. The data came in 350,000 files: one CSV file per weather station per month per year, and we had to parse an 64MB HTML file to get a link to each file. Apparently Environment Canada doesn't believe in archives (or, if they do, we couldn't find them). To download the files in a reasonable timeframe, we wrote a multithreaded Python program that used 50 threads (most threads are blocked waiting for network, so 50 is reasonable). We regularly encountered server errors, so we added resumption support to our download script. Additionally, to avoid having 350,000 files on disk, our program wrote the files directly into a `.tar` file. Overall, the raw data came in at 177MB compressed (over 2GB uncompressed).

We were hoping to give the weather data to Spark, but transferring that number of files to HDFS turned out to be unreasonable. We ended up first writing a multithreaded Python program that combined all the data for each station into a single file. The data was split into one file per station so that when we grouped in Spark, one station = one partition. We then used Spark to group our data into a much smaller dataset with average temperatures, max temperatures and average precipitation per station per season per year. We also had hoped to get a column of "max consecutive days without rain", however this turned out to be quite difficult to express in Spark (this was the reason we downloaded daily weather data rather than monthly weather data).

During this process, we also discovered the data itself was also not ideal: over the past 100 years, BC has had over 1,000 climate stations, some of which were only in operation for a few months. They also did not have consistent data: some only had temperature data, some only precipitation data, some both. We were also hoping to get wind data, but so few stations had wind data that we were unable to use it. This required some complex logic to match a `point_id` and a `year` with the closest weather station with data in a somewhat efficient manner while avoiding `NaN` s (see `data/weather/05-points-weather.py` if you're interested).

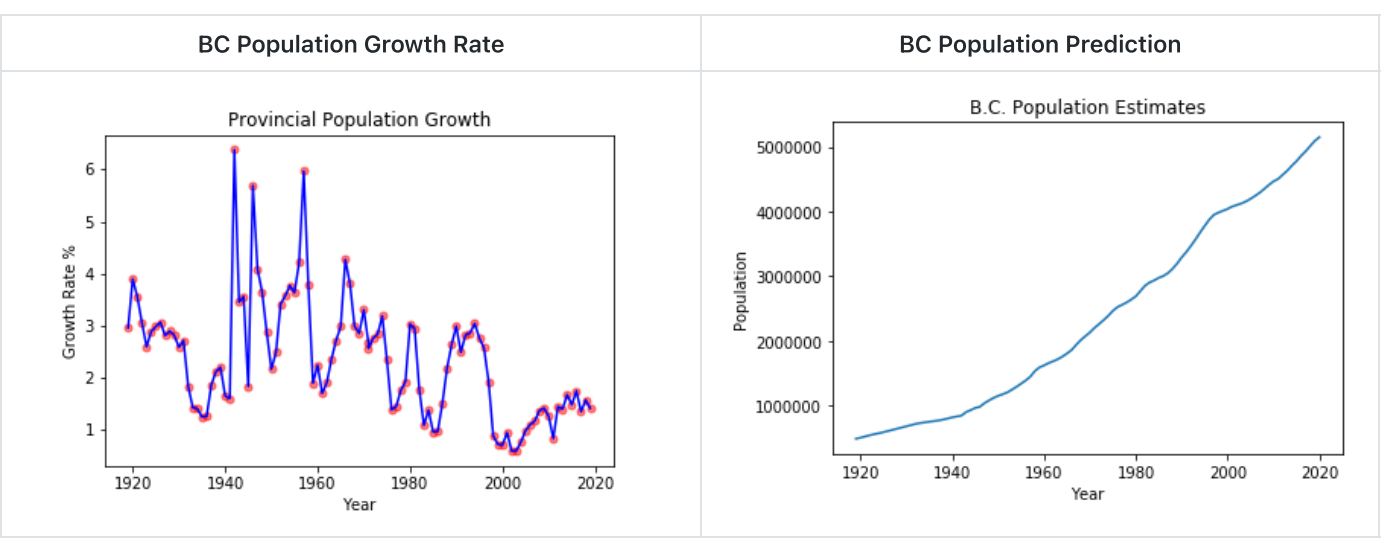
(Note that what we really wanted was historical long-range weather forecasts: when predicting fires in the future it isn't possible to use actual weather data. It turns out that data is really hard to get, so we settled for actual weather data instead.)

Population

The population data that is currently being used in the model was gathered using the B.C. Population Estimates tool. In order to get the most accurate population data for the province, we decided to use the number of people living in each local health area (LHA). We originally lookded for city-level population data but only recent population data was available. In total there are 80 LHAs throughout BC, which allowed us to give decent estimates of population density for each point.

Unforunately, data was only available from 1986-2020. However, our fire dataset contains data for 1919-2020, so we had to estimate population data for 1919-1985. We tried two different estimation methods. The first method we naively attempted was a linear regression - this method proved to not work nearly as well as we would have expected and, in some cases, resulted in a negative population, which (of course) is impossible. The second method involved using population data for the entire province, which was available for the entire time period, and calculating yearly growth rates. We then used this data to calculate the LHA population working backwards by year starting from the last available data in 1986.

These graphs are showing the data for the population growth rate that we calculated as well as the summed totals across all Local Health Areas in the BC Population Estimates.



Parks

We figured human-caused fires would likely be close to provincial parks due to campfires and negligence, so we gathered data for all provincial and national parks in BC from the BC Government. After this, we calculated the distance to the nearest park for each of the points.

Wooded?

One would think that whether an area is forested would have a significant effect on whether a forest fire can occur. No trees, no fire. Natural Resources Canada posts geometry files containing whether an area is wooded or not, and we used this data to add a boolean forested / not forested feature to each point. (It turns out this didn't help our model, more later.)

Machine Learning

Exploratory Analysis

Prior to doing any machine learning, we first did some exploratory data analysis to gain familiarity with our dataset. We noticed our dataset is quite imbalanced: there are 13,194 rows with a fire and 134,094 rows without a fire. From this, we determined upsampling / downsampling was likely required. We also paid close attention to the classification report rather than relying on accuracy score alone.

Datasets and Features

We created 4 different datasets to explore, based upon fire characteristics:

1. All Fire Causes
2. All Fire Causes, Fire Area > 1 sq km
3. Human-Caused Fires
4. Weather-Caused Fires

Our original goal was to try to predict all fires, but we also thought the model might be able to make better predictions looking at only large fires or if we split the fires by cause.

Here are all the features we collected:

```
'year', 'last_fire_year', 'lat', 'lon', 'elevation', 'park_distance', 'pop_density', 'wooded',
'spring_maxTemp', 'spring_avgMaxTemp', 'spring_avgMinTemp', 'spring_avgTotalPrecip',
'summer_maxTemp', 'summer_avgMaxTemp', 'summer_avgMinTemp', 'summer_avgTotalPrecip',
'fall_avgMaxTemp', 'fall_maxTemp', 'fall_avgMinTemp', 'fall_avgTotalPrecip'
```

When split up by cause, we changed the features we gave the model based on the addtional information we knew about the fire (e.g. population density is primarily relevant for human-caused fires). Some of these features we ended up omitting in our training (see RFECV section). We also tried replacing `year` and `last_fire_year` with a single `years_since_last_fire` feature, but ended up getting worse results (perhaps fires are cyclic and so the year itself actually matters).

Models

We tried a large number of models, including (but not limited to):

- Random Forest
- K Nearest Neighbours
- AutoGluon
- Neural Net (`MLPClassifier`)
- Gradient Boosting Classifier (surprsingly, did worse than a Random Forest)

Our initial thoughts were that a Random Forest would likely perform well, since we have a large amount of data and it would be able to ask questions based upon the most relevant / important data. However, we also thought that K Nearest Neighbours might perform well since we have geospatial data: "nearest" means something. AutoGluon auto-tunes hyperparameters, trains a variety of models and then combines them into a weighted ensemble based upon the provided metric. It was developed by Amazon AWS (not in `scikit-learn`).

Training and Tuning

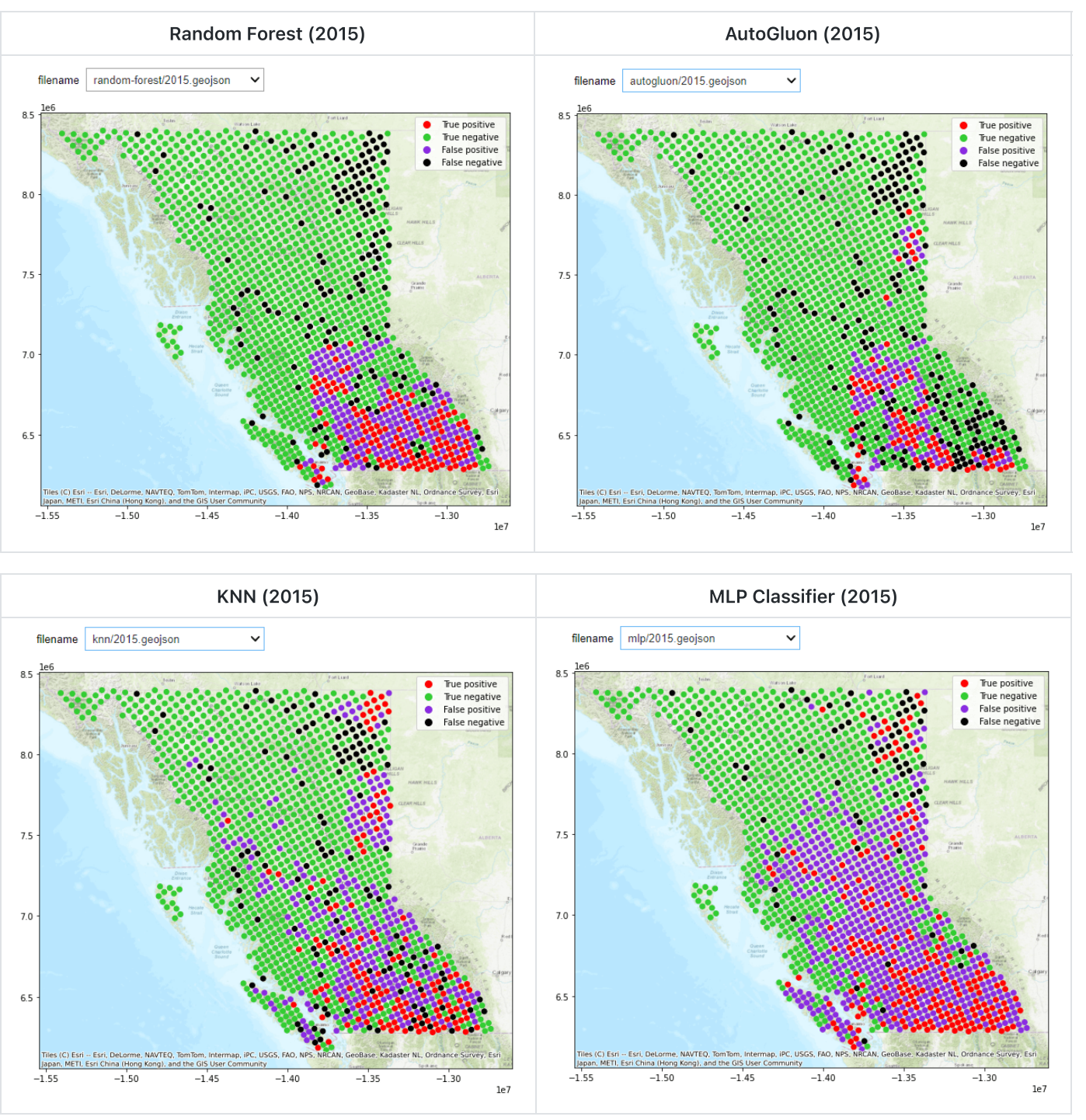
We primarily tuned model hyperparameters manually by training, visualizing the results, then making adjustments. Rather than relying on accuracy score or the classification report, both of which can be quite misleading, we found we were best able to improve the model by mapping and visualizing the predictions. We also attempted to use RandomizedSearchCV to automate this process. Rather than scoring using accuracy (the default), we instead told it to use the `f1_macro` score due to our imbalanced dataset. Unfortunately we didn't get good results with `RandomizedSearchCV` - the hyperparameters suggested ended up memorizing the training data most of the time.

For the `AutoGluon` model, both training and validation data can be provided to the model during the fit process. It uses the validation data to automate adjusting the hyperparameters and, once adjusted, then blends the data together to actually train the model. By default it will select some random validation data; however, we decided to have it train on 1919-1995 and then use 1996-2014 as validation. This validation data better matches how we're using the model and yielded better results.

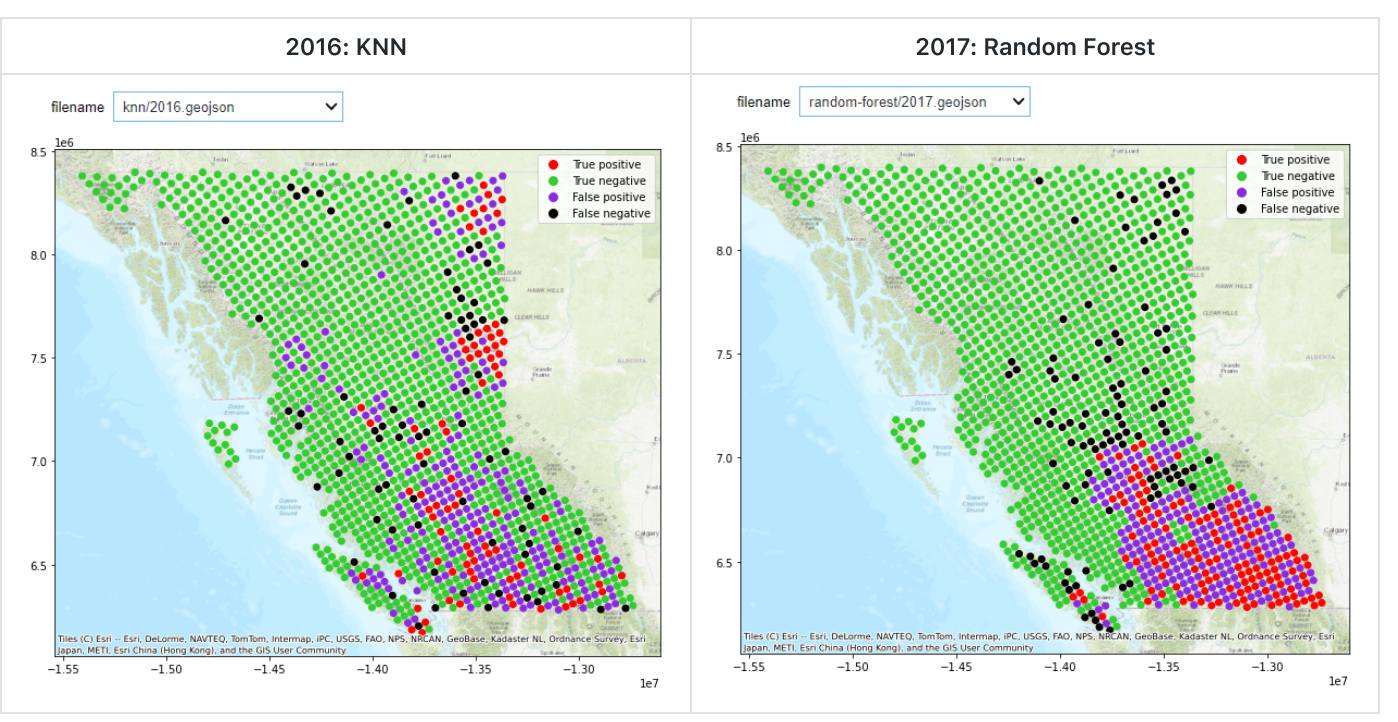
Previously it was noted that our dataset is imbalanced. We tried both upsampling and downsampling and found that, while both improve our predictions, we still were not getting good results. We then tried "partial upsampling": upsampling so that `len(fire) = 0.5 len(no fire)` and found this gave much better results.

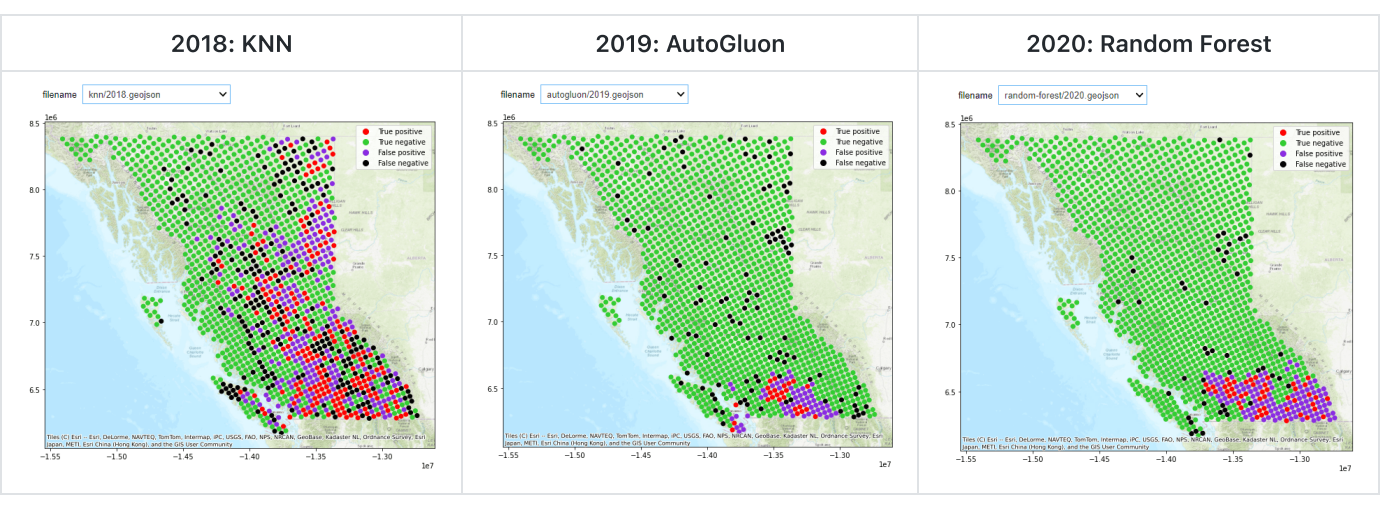
Results

We got the best results using a Random Forest and AutoGluon, although KNN did well on years with lots of fires. The MLP Classifier's output varied quite a bit each time we ran it (not enough training data), but in general it predicted fire too often. Here is the output from each model on 2015 data using all fire causes and partial upsampling. Note that green = correctly predicted no fire, red = correctly predicted fire, purple = incorrectly predicted fire, black = incorrectly predicted no fire.



Here is the "best" output for 2016-2020, although note that "best" is quite subjunctive:





The predictions were not as good as we were hoping for. However, even though each point was not predicted correctly, the models did a good job of capturing the general area where most fires would occur each year and "how much" fire (except for the `MLPClassifier`). Additionally, looking at multiple models gives a much better indication of the fires that might occur.

We got good results when examining only human-caused fires: they mostly occur in the same locations every year, so they're less "interesting" but easier to predict. Our results were not nearly as good examining only weather-caused fires - it turns out lightning is more random than humans are. For the large fire dataset, there were not enough large fires to train a model properly and we got horrible results. (Screenshots of results not presented here because our report is already quite long.)

How to Run

To reproduce our results, we have created an interactive machine learning script: `python3 learning/main.py`. It asks a series of questions, then trains a model and finally outputs predictions as GeoJSON files for the year(s) selected. These predictions can then be visualized using the Jupyter notebook found in the `visualization` directory. We have also committed some predictions to our repo that can be visualized without needing to re-run the models. Some sample script output is provided in learning/README.md.

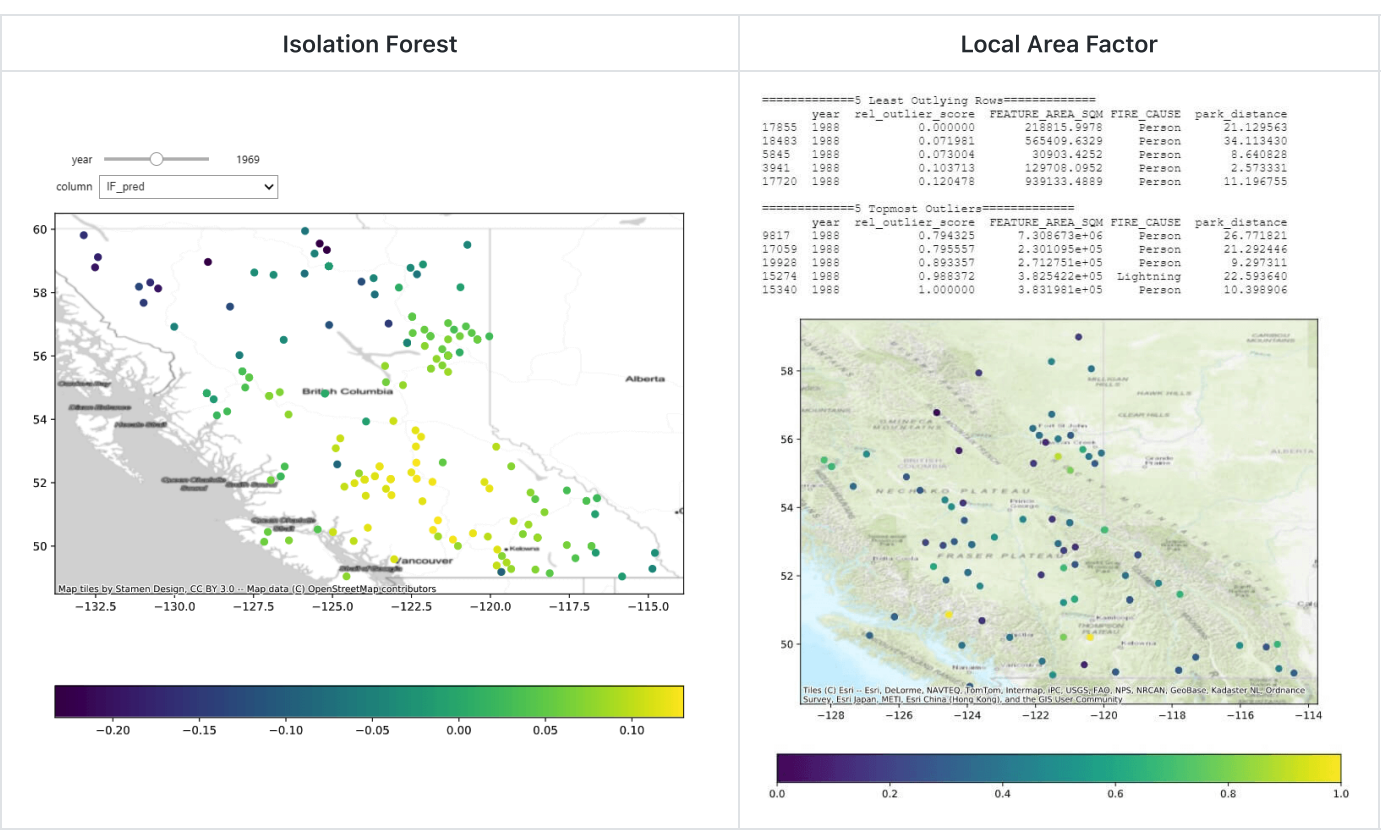
Outlier Detection

We experimented with two different outlier detection algorithms to see if we could find something "weird" common to points with fires to improve our predictions: an Isolation Forest and Local Outlier Factor. To make it easier to see the results, we created another visualization tool to showcase each model's findings (take a look at the `visualization/anomalies.ipynb` notebook if you're interested).

To make analyzing the outliers a bit easier, we started with the following columns: `latitude | longitude | fire area | fire cause` and then added additional columns one at a time to see what anomalies we could detect. Tests were conducted on `wooded areas`, `distance to parks` and `elevation`. Unfortunately we were unable to extract any concrete results - it wasn't exactly clear why certain fires were selected as as outliers.

It is worth mentioning that for some data sets the Isolation Forest focused heavily on the coordinates of the fires (see picture below). There is a clear circular pattern emitting from the center of the image: points closer to the center are yellow ("not weird"), while ones near the edges are more purple ("more weird"). The visualization really helped identify the coordinate issue; it would have been difficult to notice this looking at the data in table form.

The Local Area Factor algorithm did not have this issue but it was still not exactly clear as to how it selected outliers. From the following example, none of the 3 columns seem to be indicating any sort of human-detectable "weird" behaviour. Similar results occurred for `elevation` and `wooded` data. Although we had hoped to use these outliers to improve on our model, unfortunately we didn't get any usable results.



RFECV

As we gathered more data, we noticed training time was increasing, therefore we decided to explore how we could reduce it. One idea was seeing which columns were less important to the model and thus could be removed. We looked at two methods of analyzing column importance. Since we originally were using a `RandomForestClassifier` we initially looked at its `feature_importances_` property after training. However, since this property was specific to a Random Forest, we found `RFECV` which can be used to find important features for other models. The results of the two feature selection algorithms are presented below. These were both run on the `RandomForestClassifier` so we could compare the results. Interestingly, both seemed to agree that `wooded` data served almost no purpose for the model, perhaps because most of BC is forested despite not all burning. We used the REFCV results to create a recommended list of features that ignored `wooded` and most of the spring and fall weather data.

Random Forest `feature_importances_`	RFECV Results
Feature: year Score:0.110	Feature: year Score:1.000
Feature: last_fire_year Score:0.180	Feature: last_fire_year Score:1.000
Feature: lat Score:0.126	Feature: lat Score:1.000
Feature: lon Score:0.108	Feature: lon Score:1.000
Feature: elevation Score:0.007	Feature: elevation Score:7.000
Feature: park_distance Score:0.023	Feature: park_distance Score:5.000
Feature: pop_density Score:0.065	Feature: pop_density Score:4.000
Feature: wooded Score:0.001	Feature: wooded Score:16.000
Feature: spring_maxTemp Score:0.010	Feature: spring_maxTemp Score:9.000
Feature: spring_avgMaxTemp Score:0.008	Feature: spring_avgMaxTemp Score:8.000
Feature: spring_avgMinTemp Score:0.003	Feature: spring_avgMinTemp Score:11.000
Feature: spring_avgTotalPrecip Score:0.002	Feature: spring_avgTotalPrecip Score:13.000
Feature: summer_maxTemp Score:0.069	Feature: summer_maxTemp Score:3.000
Feature: summer_avgMaxTemp Score:0.146	Feature: summer_avgMaxTemp Score:1.000
Feature: summer_avgMinTemp Score:0.005	Feature: summer_avgMinTemp Score:15.000
Feature: summer_avgTotalPrecip Score:0.101	Feature: summer_avgTotalPrecip Score:2.000
Feature: fall_avgMaxTemp Score:0.006	Feature: fall_avgMaxTemp Score:10.000
Feature: fall_maxTemp Score:0.024	Feature: fall_maxTemp Score:6.000
Feature: fall_avgMinTemp Score:0.004	Feature: fall_avgMinTemp Score:14.000
Feature: fall_avgTotalPrecip Score:0.003	Feature: fall_avgTotalPrecip Score:12.000

Stats

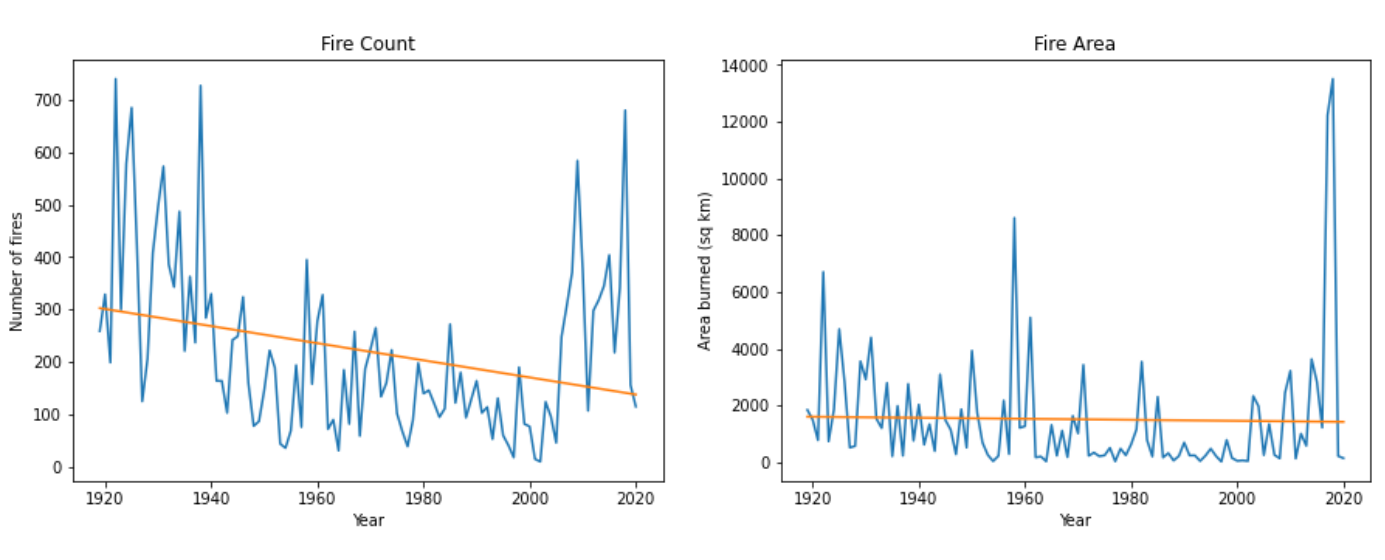
(See `stats/change-over-time.ipynb`)

While our primary goal was to use machine learning to predict fires, during our research we came across many news articles claiming fires are getting worse each year. BC's Premier recently blamed climate change for BC's recent wildfires. The federal Environment Minister claimed the same thing and went on to say, "all of **the data** suggests the extreme weather won't be improving in the future" (emphasis ours) however no specifics were given in the article.

Have wildfires in BC been getting worse (changing over time) - does the data back up their claims? We explored the following two questions using BC Wildfire data from 1919 - 2020:

1. Have the number of wildfires per year changed?
2. Has the total area burned by wildfires per year changed?

To answer these questions, we performed two linear regressions: one on the number of fires per year and one on the total area burned per year and were looking for `p < 0.025` (`0.05` with Bonferroni correction). We found that the number of fires has actually been decreasing (`p = 0.0027` + subsequent one-tail test) and did not come to a conclusion about total area burned each year (`p = 0.8115`). The data does not seem to support the claims being made by the Premier and the federal Environment Minister (at least the data we have collected).



Conclusion

Being able to accurately predict wildfires could save lives and a significant amount of money. While we got decent results, we don't believe our results are good enough that they would be actionable (e.g. actually reallocate fire resources based upon the models). If we had more time, we would have liked to try to combine the results from our best models to see if we could have further improved our predictions and we also had fire area data that we would have liked to run a few regression models on to try to predict how big a fire might become.

Accomplishment Statements

Ethan

- Generated geospatial data using Geopandas and spatially joined it with other datasets, allowing for further analysis with machine learning tools
- Tuned a `RandomForestClassifier` to predict future wildfires in BC by hand and using `RandomizedSearchCV`, resulting in an accuracy score of 0.77 on 2015 data
- Cleaned and transformed 350,000 weather data files (2GB) efficiently using the Python `multiprocessing` library and Apache Spark, allowing the data to be used as a machine learning feature
- Analyzed statements made in the media about wildfire frequency using statistical tests and concluded the statements may not be accurate

Mikhail

- Gathered local health area population data and calculated population estimates from 1919 - 1985 using province wide population growth statistics to determine whether human factor could be used as a predictor for future fires.
- Retrieved BC park information data from the BC Government and determined the closest park to each of the grid points using geopandas in an attempt to determine how the placement of parks can affect a chance of fires.
- Created outlier detection models, specifically using Local Area Factor and Isolation Forest algorithms to determine if any anomalies could be located in the dataset to prevent them from confusing the machine learning models.
- Created interactive visualization tools in jupyter notebooks with the help of ipywidgets to show the results of various machine learning models in a user-friendly form.