Can we predict fire with weather data?

Harsh (hp444), Yi Yao (yy899), Murali (mt788)

December 10, 2019

Contents

1 Introduction

The weather has literally been the hot topic of discussion in recent times. One aspect of weather data is that, when it is combined with other data sets, it can be used to predict varied different aspects related to the society. For example, one particular aspect could be how does traffic conditions change based on the weather in a particular city or if there is a correlation between crime rates and weather conditions. In this project, we are trying to address a more important issue of predicting wildfires in different cities based on weather conditions. Since the future weather data is readily available these days, a good model would help predict these wildfires in advance and take necessary precautions to completely avoid it. This would help avoid human and wildlife loss. It would also help contain the impact of these wildfires on the environment which increases the content of poisonous gases in the environment, making the nearby uninhabitable.

Wildfires are growing in frequency and intensity by the day. The 2019 wildfire season of California has more than 6800 fires recorded by the US Forest Service and approximately 250000 acres of burnt land. These wildfires could be the result of foul human activities but there has been a lot of speculation on the high correlation of these fires with the weather. Climate change is not only making the fire season longer but on average much more intense. The idea is to use the model to predict these fires in advance, thereby reducing such incidents.

2 EDA

We have created a website to help visualize our dataset.

`https://leafyao8621.github.io/firevisualization.github.io/`

2.1 Covariates

To build our final dataset, we wrote a python script to get the geographical coordinate (latitude and longitude) of all cities and built tables with missing information about fires. Then, we went through the database with information about fires and filled in the fire indicator in the tables using the coordinates obtained in the previous step.

The features that we have in our final dataset are:

Table 1: Features

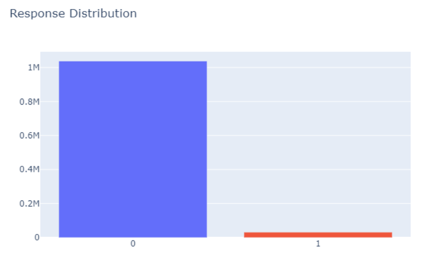
Covariate	Type	Description
Humidity	Real	Humidity at a given hour
Pressure	Real	Atmospheric Pressure at a given hour
Temperature	Real	Atmospheric temperature at a given hour
Weather Description	Categorical	Description of weather at a given time, e.g. “sky is ckear”
WindDirection	Real	Wind direction in degrees at a given time, North being 0°, clock-wise being positive direction
WindSpeed	Real	Wind speed at a given time

Table 2: Response

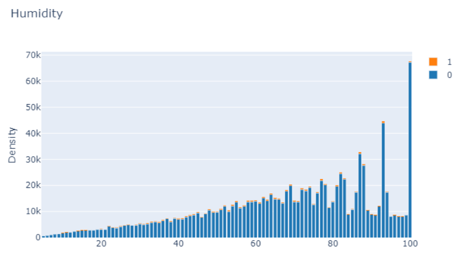
Response	Type	Description
Fire	Boolean	Whether there’s a fire at a given hour

2.2 Basic Statistics

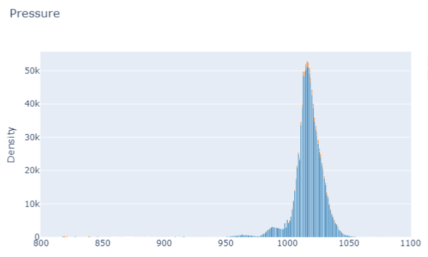
The size of the dataset in the Fires table is: $18,800,465 \times 39$, the size of the weather table is: $1,629,108 \times 10$ and the size of the final table is: $1,629,108 \times 11$. The following figures help illustrate what the final dataset for all of the cities looks like:



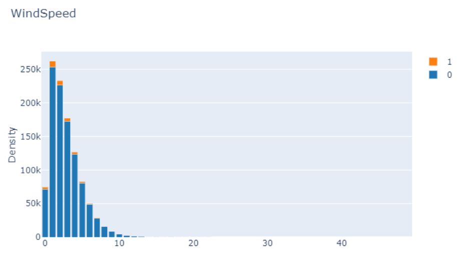
(a) Response Distribution



(b) Histogram of Humidity

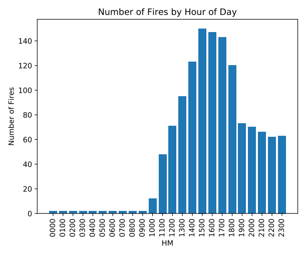


(c) Histogram of Pressure

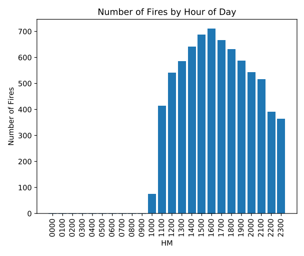


(d) Histogram of Wind Speed

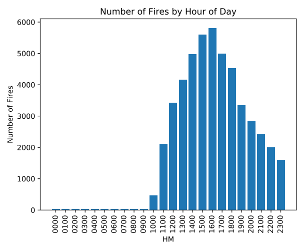
Figure 1: Visualization



(a) San Francisco

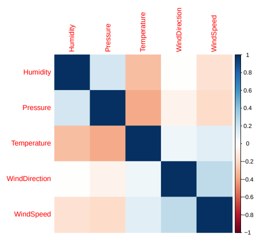


(b) New York

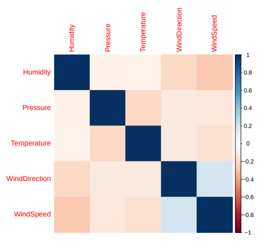


(c) All Cities

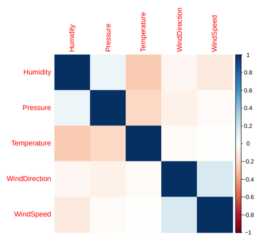
Figure 2: Hourly Fire Distribution



(a) San Francisco



(b) New York



(c) All Cities

Figure 3: Correlation Plot

From Figure 1, we observed that we have significantly fewer positive samples (data points where the fire indicator has value 1) than negative samples (data points where the fire indicator has value 0).

From Figure 2, we observed that very few fires are recorded in the time span of 00:00 to 10:00, while almost all fires are recorded in the time span of 10:00 to 23:00. We have realized that the negative samples in that time span might be due to missing data, or people not reporting during that time span. If we fit a hourly seasonality model, we may overfit to data points in this time span, and end up predicting no fire in this time span, potentially making our models Weapons of Math Destruction.

From Figure 3, we observed that the correlation between features is DIFFERENT in different cities, as well as overall, which justifies our decision to build city-specific models.

3 Brief Description of Algorithms and Definitions Used

3.1 XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance.Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. A popular example is the AdaBoost algorithm that weights data points that are hard to predict. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

3.2 SVM

A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection.[3] Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier.

3.3 Decision Trees

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

3.4 DBScan

DBSCAN - Density-Based Spatial Clustering of Applications with Noise. DBScan finds core samples of high density and expands clusters from them. It is good for data which contains clusters of similar density.

3.5 Boosting

Averaging predicted values from estimators trained on balanced random subsets of data. For example, if we have a data set with 10 positive samples and 100 negative samples, we train the estimators on multiple balanced random sub-samples of the dataset, each containing all the positive samples. The n_boost parameter determines the number of such subsets.

3.6 Balanced Accuracy

The balanced accuracy in binary and multiclass classification problems to deal with imbalanced datasets. It is defined as the average of recall obtained on each class.

3.7 Metrics

The training metrics are 20-fold cross-validation metrics while test metrics are metrics on a separate balanced test data set that no estimator is trained on.

4 Model Building and Accuracy Tuning

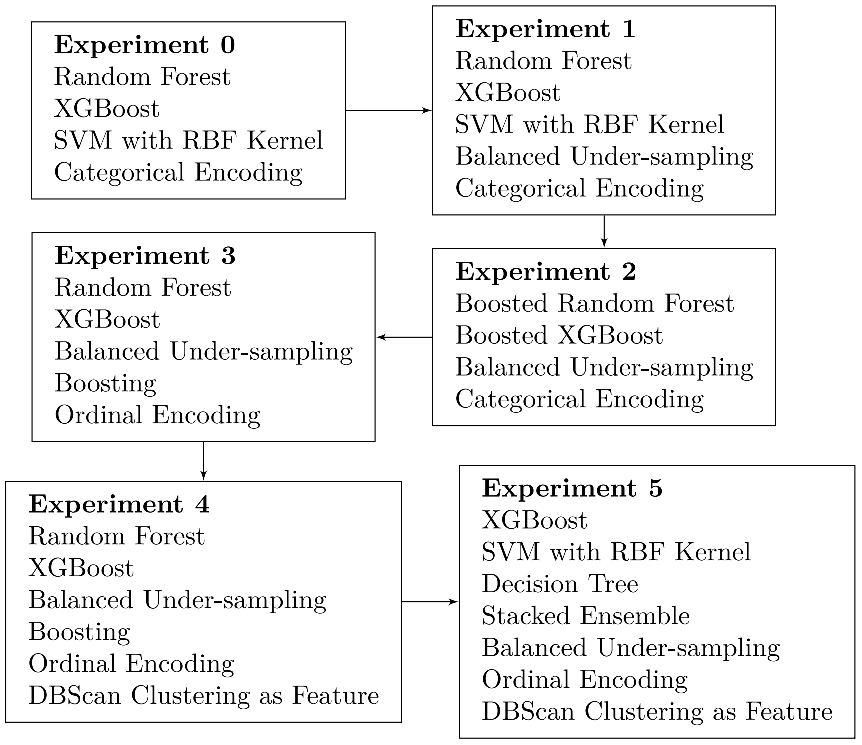


Figure 4: Flowchart of Experiments

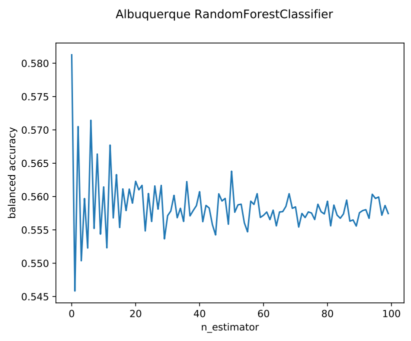
4.1 Preliminary Analysis

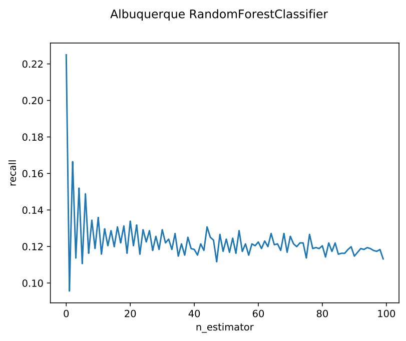
Table 3: Training and Test Metrics for Albuquerque

Experiment	Classifier	Meta Parameter	Training	Test
0		n_estimator	bal_acc	bal_acc
	XGBClassifier	96	0.5167	0.5217
	RandomForestClassifier	1	0.5813	0.6591
1		n_estimator	bal_acc	bal_acc
	XGBClassifier	100	0.6654	0.6509
	RandomForestClassifier	82	0.7039	0.6656

4.1.1 Experiment 0 - Primitive Model Fitting

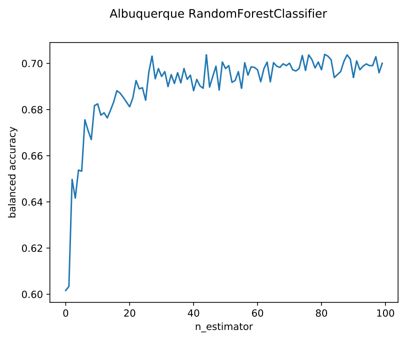
First, we fit XGBoost, Random Forest and SVM by using the entire dataset for each city and found the accuracy to be quite low. The low accuracy could be because of the model overfitting. There are, overall, 1M negative samples and roughly 25k positive samples.

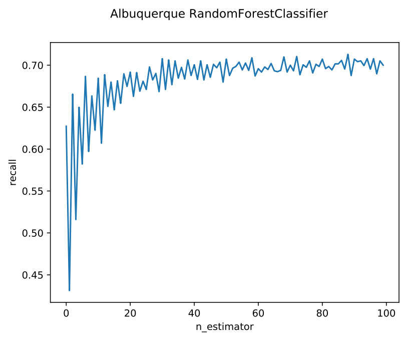




4.1.2 Experiment 1 - Under-sampling

Recognizing the unbalanced nature of our dataset, we created a balanced training data set by combining all the positive samples with a random subset of all the negative samples. Using this balanced training data set, we fit XGBoost, Random Forest and SVM with RBF kernel. We observed a significant improvement in test accuracy in the XGBoost and Random Forest models; however, we have not observed any significant improvement in the SVM models.





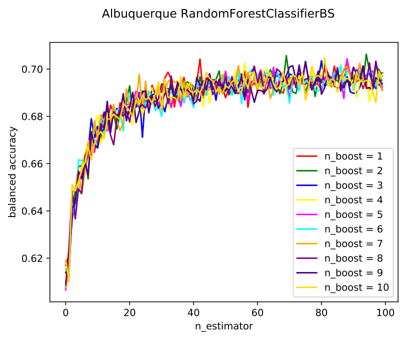
4.2 Accurarcy Tuning

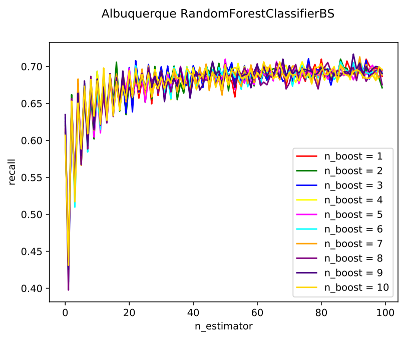
Table 4: Training and Test Metrics for Albuquerque

Experiment	Classifier	Meta Parameter		Training	Test
2		n_boost	n_estimator	bal_acc	bal_acc
	XGBClassifierBS	3	100	0.6636	0.6523
	RandomForestClassifierBS	2	95	0.7063	0.6911
3		n_boost	n_estimator	bal_acc	bal_acc
	XGBClassifierBS	4	96	0.6668	0.6551
	RandomForestClassifierBS	3	89	0.7031	0.6935
4		n_boost	n_estimator	bal_acc	bal_acc
	XGBClassifierBS	7	88	0.6634	0.6511
	RandomForestClassifierBS	5	74	0.7011	0.7035
5			n_ensemble	bal_acc	bal_acc
	EnsembleClassifier		90	0.7185	0.7080
	XDSClassifier		96	0.7138	0.7327

4.2.1 Experiment 2 - Boosting Models Fitted with Balanced Sub-samples

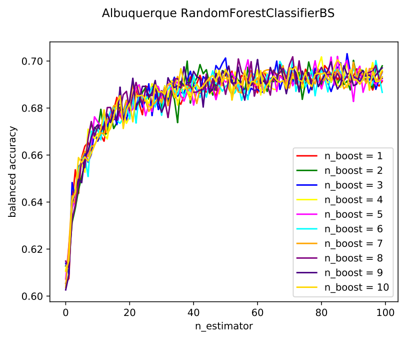
Having the possibility of underfitting due to under-sampling in mind, we have experimented with boosting to increase accuracy. By keeping all the positive samples and randomly resampling negative samples equal to the number of positive samples, we have kept the balance of the training data set while making better use of the data set we have available.

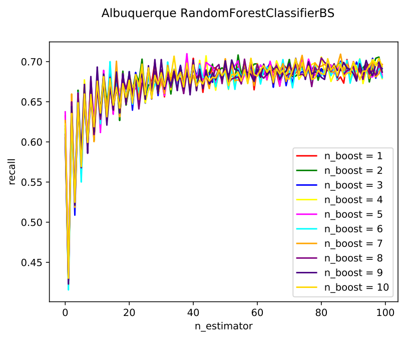




4.2.2 Experiment 3 - Clusters of Many-hot Encoding

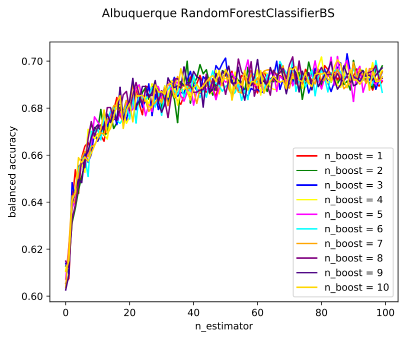
Having observed some order in the covariate “Weather Description”, we have experimented with many-hot encoding for ordinal variables. Since an overall order could not be established, we have turned to many-hot encoding on 3 clusters of levels and one-hot encoding on the remaining levels. As for models, we continued to use the boosted algorithms in the previous experiment.

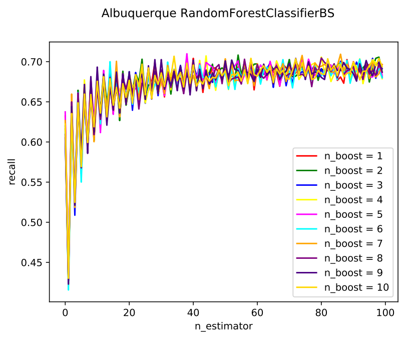




4.2.3 Experiment 4 - DBScan Clustering as Feature

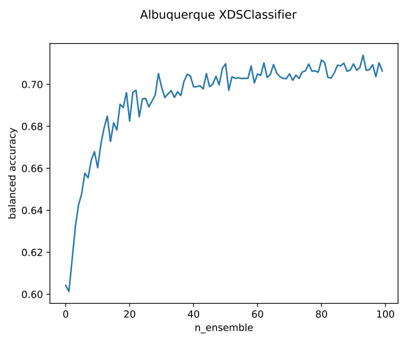
We have also experimented adding one-hot encoded output of an unsupervised clustering algorithm, DBScan as covariates to the dataset to be fitted with supervised models. As for models, we continued to use the boosted algorithms in the previous experiment.

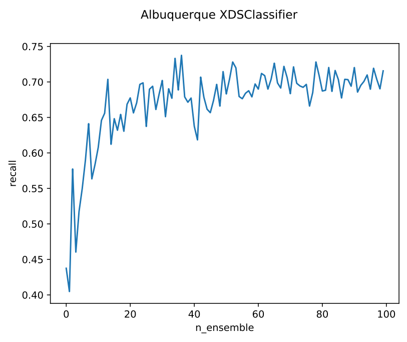




4.2.4 Experiment 5 - Stacked Ensemble

In this experiment, we explore feature engineering through the addition of two features, one which is to use the weather types as oridinal features and another one by grouping them using DBScan Algorithm and using their groups as features. We tried to utilize the power of ensemble learning where we combine various weak learners in order to make better predictions. The algorithms that we picked are SVMs with RBF Kernel, Decision trees and XGBoost. These were identified to perform poorly with the original dataset fed as a whole.





5 Conclusion

Overall, our objective was to be able to predict fires given weather data. Our accuracy depended upon the methods we picked to push the accuracy up. We identify that we hit the information "roof" when it comes to how well we could predict the fire. Some other interesting observations are as follows:

* The XGBoost algorithm when fed with one round of undersampling with the no. of estimators higher than 25 seemed to have learned all the information available.
* The stacked ensemble algorithm performed the best for the cities shown in this report. It's interesting to note that although weak learners do not capture complex information, when we differ the way they capture that information, we see that they perform well collectively without having to weigh it in the final prediction.
* The information roof that we clearly hit which may indicate that we needed more real features that inherently affect the prediction. That may be the error that we seem not able to avoid.
* Incorrectly predicting that a city will not have an occurence of fire may lead to unpreparedness on the part of the fire services which will further cause data points to pop up as "fire" until our model really learns that there are a lot of fires. Our project may in this sense a weapon of math destruction. This problem can be overcome by including many more features, more intelligible data and bumping up the accuracy to an acceptable level.
* Since we have built a complete tool-chain from data-cleaning to automated parameter selection, we are willing to use it in production given that the company continues to use the same data sources.

References

[1] Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. The balanced accuracy and its posterior distribution. *ICPR '10 Proceedings of the 2010 20th International Conference on Pattern Recognition* (2010), 3121–3124.

[2] Nori, H., Jenkins, S., Koch, P., and Caruana, R. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223* (2019).

[3] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[4] Wikipedia contributors. Decision tree — wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Decision_tree&oldid=929548230`, 2019. [Online; accessed 9-December-2019].

[5] Wikipedia contributors. Support-vector machine — wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=928737848`, 2019. [Online; accessed 9-December-2019].