Machine Learning Algorithms for Big Data with Python

Building a Hiking Partner Recommender System

Grace Mills and Tate Rosen, Research Director Dr. Kevin Treu

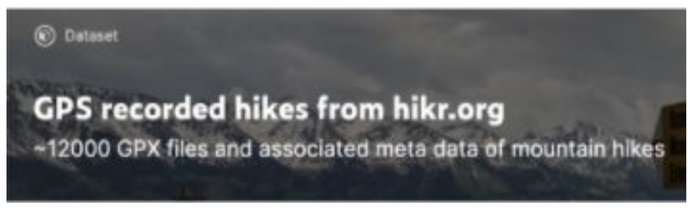
MACHINE LEARNING

Many different machine learning algorithms exist and can be applied to data to reach conclusions for different research questions. Machine learning can be supervised or unsupervised, meaning the class that is trying to be learned is either unknown or known. Machine learning can be used to classify new instances of data or associate types of data with one another by making rules. We experimented with many classification algorithms before choosing which ones to implement on our dataset.

PYTHON LIBRARIES

Python contains a wealth of different libraries and packages that can be used when implementing different machine learning models. One such package, Numpy, provides mathematical functions to manipulate data. Pandas, another library, gives the user the ability to load data into a dataframe, along with many other functions. Scikit-learn is the main machine learning library for python, and Matplotlib allows the user to create plots and visuals with the data.

HIKING DATA

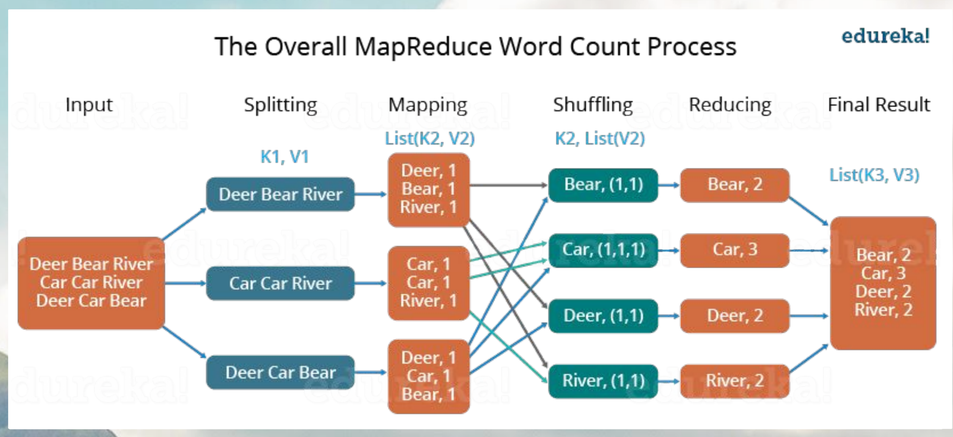


Kaggle.com is an online website where many datasets and coding examples are posted for public use. We explored different datasets and topics from this website and ultimately decided to focus on a dataset about hiking, which included many attributes like length and elevation. Next, we preprocessed the data by removing outliers, selecting the most predictive features, normalizing attribute values, and balancing class values. After difficulties using Numpy, we finally decided to use the Pandas library on this dataset.

FURMAN
COMPUTER SCIENCE

This research was supported and funded by the Department of Computer Science, the Kresge Institute, and the Furman Summer Fellows Research Program.

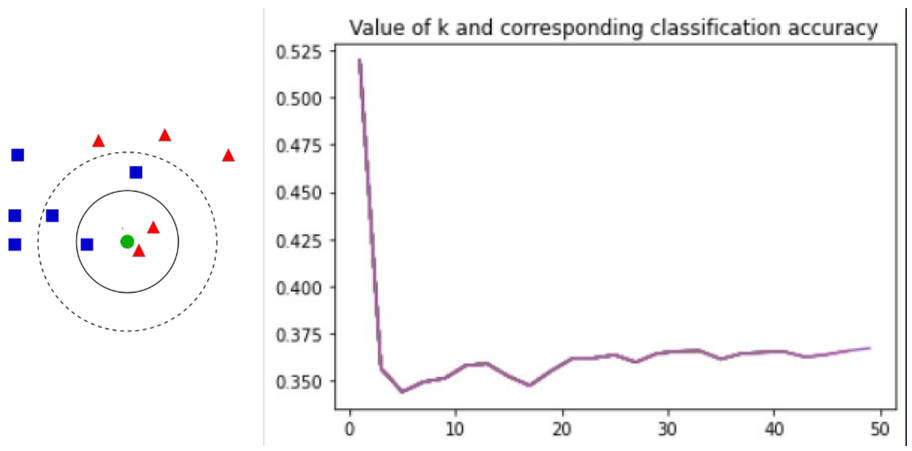
MAPREDUCE/AWS



MapReduce is a programming paradigm that allows that performs parallel processing on large datasets. MapReduce can be implemented easily in Python. A simple example of a MapReduce program is a word counter. Text data is input into a mapper which yields a key-value pair with each word given and a 1. The mapper passes on ("Word", 1) to the reducer. Then, the reducer's job is to sum up the number of times each word appeared. The output from this MapReduce program would be a list of key-value pairs where the key is a word and the value is the number of times it appeared in the data. For example, if "The" appeared 5 times in the dataset, the key-value pair output by the reducer would be ("The", 5).

Amazon Web Services, or AWS, has many useful features and allows MapReduce programs to be run on multiple computers. Running programs in parallel using services like AWS allows them to provide results in less time when using a large dataset.

KNN



Nearest neighbor classification, or kNN, uses the instances of data closest to the current instance for a specific attribute to classify the current instance into a certain category. We used the Minkowski distance metric when implementing this algorithm on our hiking data and held out 30% of the data for testing. This method only provided about 50% accuracy in predicting difficulty levels, but can we do better? Further research could help improve these results.

RECOMMENDER SYSTEM

```
def steps(self):
    return [
        MRStep(mapper=self.mapper_parse_input,
               reducer=self.reducer_attributes_by_diff),
        MRStep(mapper=self.mapper_create_user_pairs,
               reducer=self.reducer_compute_similarity),
        MRStep(mapper=self.mapper_sort_similarities,
               reducer=self.reducer_output_similarities)]
```

↓

```
"olethros"      ["cristina",0.9787364341907547,1074]
"pizzo1954"     ["cristina",0.9744371501881771,780]
"cristina"      ["olethros",0.9787364341907547,1074]
"cristina"      ["Abominevole",0.9822285661501574,512]
"Marchino"      ["cristina",0.9716312526748905,768]
```

Our main goal was to create a hiking partner recommender system that would take input containing different hikes and output two users that would make good hiking partners. To accomplish this, we extracted usernames and several hiking attributes, including length, uphill distance, downhill distance, maximum speed, and moving time, from the dataset and used a MapReduce model to determine similarity between users. This model incorporated three different pairs of mappers and reducers to eventually reach a conclusion about which hikers were most similar. We included the cosine similarity function in one of these reducers to determine these levels of similarity.

```
for lengthX, lengthY in lengthPairs:
    sum_xx += np.dot(lengthX, lengthX)
    sum_yy += np.dot(lengthY, lengthY)
    sum_xy += np.dot(lengthX, lengthY)
```

CONCLUSIONS

Not every dataset is fit for 'good' data mining. In the hiking dataset we used, the correlation between attributes was very low, so the results did not yield a very high accuracy. Also, not every data mining problem can be improved or solved using MapReduce. While MapReduce can yield better and faster results this is only true when using *extremely* large datasets.

FUTURE WORK

To continue this research, we would like to experiment with different machine learning algorithms on the hiking dataset such as kNN Regression, Naïve Bayes, or Decision Trees. We would also like to implement our recommender system in AWS, instead of just locally, to improve run time. Finally, we would like to experiment with different parallel processing frameworks like Spark, an open-source unified analytics engine for large-scale data processing.