Big Data Parallel Programming
Project

Fredrik Mårtensson

May 21, 2020

Contents

1 Introduction

As a part of the course *Big Data Parallel Programming* at Halmstad University a task for processing and analyzing big data is requested through the use of *Apache Spark*. The project is based on information given from the course Parallel Processing Systems for Big Data: A Survey[1], Mining of Massive Datasets [2] and Computer architecture: a quantitative approach [3]. The project focus on extracting Severity from US accidents shared by *A Countrywide Traffic Accident Dataset* [4] and *Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights* [5]. The dataset consists of a total of 2925212 rows and 47 columns which include information such as coordinates, position, time, weather conditions and location information. The task is to predict Severity of accidents in different areas of the United States and being able to provide an overview of where improvements are needed to increase traffic safety.

In background the different platforms and libraries will be presented as well as previous research on the field of big data. The method contains an explanation of the steps for analysis and preprocessing that have been carried out. In the result section a result is given based on the analysis and preprocessing and finally a discussion and conclusion are given that analyse the presented result. The evaluation and models is created through Jupyter labs.

2 Background

In recent years it has become increasingly popular to utilize parallelization to efficiently multitask and utilize the performance for data analysis. Spark works very well for preprocessing and could be complemented to tensorflows ML library in order to create efficient and powerful operations. The difference between spark and pandas depends on the dataset. Pandas works better for small sets but not as great when the size is greater than the memory given to the process.

Cloud computing has become an increasingly popular topic for big data. Cloud computing allows companies to rent cloud services in order to reduce maintenance costs and spend more time processing and managing information. Services such as IBM cloud, Microsoft Azure and Google Cloud are companies that rent out cloud-based services and enable data management through the Jupyter labs for python and Spark over the internet. This project consists of using IBM cloud to analyze, preprocess and perform machine learning on data.

2.1 State-of-the-art

According to an article on *The rise of big data on cloud computing: Review and open research issues* [6], it describes how Cloud computing has become a standard for large-scale and complex computing by utilizing parallel processing, security and service interaction with scalable data storage. This adds lower infrastructure maintenance and costs to businesses. Companies offering cloud computing services include IBM, Microsoft Azure, Google and Amazon AWS.

According to *Big data processing in cloud computing environments* [7], it exists multiple different distributed file systems as Google File System (GFS) and Hadoop Distributed File System (HDFS) which is an open source version of GFS. According to [6] HDFS can run on top of local file system of cluster nodes and store large files suitable for streaming data. It provides faulit tolerance and scaling from one to thousands of nodes. The architecture is built upon a master and slave relationship. D. Kossmann et al.[8] presents four different architectures which is based on multi-tier database application architecture including partitioning, replication, distributed control and caching architecture. This architecture can be used in order to handle large data processing models that commercial database management system (DBMS) are not suitable for. In order to acheive better scalability and performance when processing big data,

parallelization techniques and algorithms is applied. MapReduce is a popular algorithm proposed by Google that is based on GFS and adapted through Hadoop. MapReduce major advantages is that it hides details related to data storage, distribution, replication, load balancing and simplifies for developers by only specifying two functions: the map and reduce function. Multiple projects utilizes MapReduce including Spark[9], Hive[10], YARN[11] Alternative models for handling large amounts of data are provided by RankReduce [12], which combines Local Sensitive Hashing and MapReduce to implement KNN in high-dimensional spaces.

Performance Prediction for Apache Spark Platform [13] explains how Spark is used as a distributed data processing platform that utilizes distributed memory to process large volumes of data in a efficient approach. The report gives an introduction till how Spark operates, how job execution of tasks is carried out and how memory usage affect the performance.

2.2 Setup

The project utilises *Apache Spark 2.x Cookbook* [14] to optimise and streamline model creation through a docker container on local computer to improve memory and garbage collector (GC) performance. Additional google crash course and course material is used to give an overview on what theoretical operations that should be used to produce reasonable preprocessing steps, both for clustering and for classification. Python 3.6 is used combined with Spark 2.4.5 in order to work with both IBM cloud and a private docker service running all-spark-notebook[1] on the cloud. The environment used was Jupyter labs which is an extension of Jupyter notebook for additional features in development. The settings for IBM was the free usage plan with Python 3.6 and spark.

[1] `https://hub.docker.com/r/jupyter/all-spark-notebook/`

3 Method

The project is structured in five categories that can be observed in table 1.

Steps	Method
1.	Analysis
2.	Preprocessing
3.	Feature selection
4.	Machine learning
5.	Model evaluation

Table 1: Project structure

3.1 Analysis

In order to give as understanding the following data were observed. The dataset covers 49 states of the US. The dataset is based on open source from a report on A Countrywide Traffic Accident Dataset[4] and Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights.[5]. The severity level between 1-4, where 1 indicates the least impact and 4 is high impact on traffic.

The project started by conducting a comprehensive analysis of the data. To gain an understanding of the information from the dataset, some interesting data is presented, for a deeper insight check the dataset or the project analysis. The data includes which types of data that exists (string, double, bool). Then how many labels, classes and numerical values that exists. The total set consist of 2,974,335 rows, the columns is divided into one label, 28 classes and 12 numeric columns, as well as six columns deleted which included data not to be processed as text analysis and unique identifiers for each data point. Observations were made that several columns contained no value, for example. Precipitation (in) contained 1998358 missing values, Number contained 1917605 missing values. These needed to be either antedated or reduce the number of rows, but since a larger majority was missing, it was decided to remove the columns completely.

Severity	Weather conditions	State	Missing values
1 (0%)	Clear (808171)	CA (653172)	Precipitation (in) (1995584)
2 (67%)	Mostly cloudy (405136)	TX (296044)	Number (1880955)
3 (30%)	Overcast (382479)	FL (220711)	TMC (678948)
4 (3%)	Fair (311864)	SC (146340)	Wind_Speed(mph) (439231)

Table 2: Oversight of data. The parentheses containing the observed data point and number of occurances of the category.

The highest probability of accidents occurred around 2-4 during the night and at least around 16-22, figure 1. In table 2 additional information on the top occurrences of data is displayed. In figure 2 a pie plot of the Severity can be observed. In figure 3 the distribution of severity in each state is observed to give a closer look on what should be expected as result from the machine learning.

Analysis	Description
Correlation	Variable correlated with each other.
Severity distribution	How many sample of each label existed
Distribution over each state	Accident in each state
Distribution of severity over each state	Severity of accidents in each state
Weather conditions	Condition of each accident
Time of accident (Hour)	At which hour most accidents occur
Time of accident (Month)	At which month most accidents occur
Map distribution of accidents	A map that shows the distribution of accidents

Table 3: Which form of analysis that were used for the project.

3.2 Preprocessing

During preprocessing, methods were used for filters, select, groupBy, persist, unpersist and resilient distributed dataset (RDD). Several of the methods used improved performance in comparison of using pandas but persist and unpersist stood out among the crowd. By storing a data set in memory it was able to communicate between operations and according to Apache Spark's documentation it can speed up operations as much as 10x. Table 4 gives an overview of what is done during the preprocess.

Method	Description
Select state	Select a specific state if wanted
Modify time	Process timestamp to categories of hours and month
Quantilization and clustering	Quantile/cluster data based on coordinates
Cast/extract types	Cast data to correct type and divide into different variables
Clean data	Remove data with less occurance than 1%
Remove categories	Remove categories if unique value is 1
Pipeline	Imputer, MinMaxScaler, StringIndexer, one-hot encoding
Combine features	Create one single feature vector of data

Table 4: The process of cleaning and working with the data before machine learning.

Given that one-hot encoding is used, the feature vector must not be too large since it can cause memory issues during machine learning even if Spark is applied. In order to reduce the vector unnecessary data that does not contribute to predicting the target value should be excluded. Since the number of features increases more than desirable when models focus at street level, clustering is applied for both longitude and latitude feature. The alternative is to implicitly combine Zip Code, Country, Airport_Code and City to reduce the number of categories without losing larger amounts of information since Airport_Code and zip Code does not provide any additional knowledge to the feature vector. The choice of the discard streets column is based on the fact that it contains too many unique values (160715) and generates to many unique features that can cause the model to overfit. The same is given by Zip Code which contained 377152 unique values. By applying clustering the feature vector required to describe a location is reduced, instead of using categorical values for state and street the coordinates were clustered into a new feature named position. With the help of clustering it is possible to focus on smaller locations, for example states to produce a better model that divides the positions even thinner to increase accuracy. As an example with clustering with a k value of 250. These centroids should be divided and to match the data. When specifying a smaller set of data it can distinguish the locations more accurate. Alternative is to increase the k value which would imply a bigger feature vector.

During preprocessing, it is also possible to filter states by using filter to gain a better understanding of how the model performs on smaller sets of data and shorten the processing and modelling time. In order to select all states the parameter is configured to 'ALL', otherwise states is defined as example: 'CA', 'TX', 'FL'.

Timestamp for when accident occurred and timestamp since last weather lookup was preprocessed and extracted as hour and month features. Since there is different timezones in the US the timestamp was converted to cor-

respond to the correct timezone. Based on the analysis it was decided that from "Weather frequency", figure 4, all scenarios below 1% would be removed with groupBy to minimize the number of unique classes. Columns containing only one feature were also discarded since they do not provide any valuable information to machine learning.

Prior to machine learning, four steps of preprocessing were carried out including all numeric values being processed with imputes with the strategy median. For the mathematical calculation of normalization, MinMaxScaler[2] is used where min (0.0) is the lower limit and max (1.0) is the upper limit For the categorical values, StringIndexer and OneHotEncoderEstimator were used to create a vector of features. In addition to preprocessing the numerical and categorical features they need to be combined into a final feature vector which is processed using VectorAssembler.

3.3 Quantiles and clustering

Two methods for grouping data was tested, quantiles and clustering. Quantiles is based on a theory from google crash course[3] for machine learning. This means that the data in the columns Start_Lat and Start_Lng needed to be divided into equal quantities by QuantileDiscretizer, then indexed with StringIndexer and finally one-hot-encoded. This gives us two vectors for coordinates. Then cross product is applied to create a new vector that describes surfaces according to feature crossing[4]. However due to limitations between one-hot-encoding and feature crossing, which resulted in an overflow in the Spark heap, the second solution was to apply clustering through K-means (KMeans) and do a machine learning evaluation which would reduce the number of positions but increase performance. In figure 5 the different centroids and the corresponding data point is displayed. The figure is placed upon the state CA and one color can display different cluster due to that the total number of centroids is 400.

3.4 Feature selection

The features that are relevant were used to determine the principal component analysis (PCA) and ChiSqSelector. PCA [15] is used to capture essential data

[2] https://spark.apache.org/docs/2.1.0/ml-features.html#minmaxscaler

[3] https://developers.google.com/machine-learning/clustering/prepare-data

[4] https://developers.google.com/machine-learning/crash-course/feature-crosses/crossing-one-hot-vectors

patterns which can be used to determine how many features are needed to describe over 90% of the dataset. This is done by utilizing the variance of each feature. ChiSqSelector [16] is used to select which features are most relevant within the k value limit that is extracted from PCA. This is done to reduce the machining time of the machine learning and hopefully remove unnecessary information. According to table 5 it is possible to see how the feature vector size is reduced and how much data it covers according to PCA.

Pior-FV	**Post-FV**	**Prior-Rows**	**Post-Rows**	**PCA**
5593	250	2974335	2819074	97.04%

Table 5: How the models performed and how the feature vector (FV) changed and how the number of rows were reduced when using State parameter as 'ALL'.

3.5 Machine learning

For machine learning, 80% of the computer was used for training and 20% for testing. The models are evaluated through MulticlassClassificationEvaluator and Crossvalidator from Spark by testing several different grid parameters and selecting the best model from Logistic regression, Decision tree and Random forest. Each model present a model and evaluated through accuracy based on the cross-validation and a confusion matrix which gives the true and predicted values of the model.

4 Result

The tests for the models were carried out with the same parameters for everyone to give a better overview of how the models performed. Logistic regression used the following parameters regParam: [0.1,0.01], maxIter: [10], elasticNetParam: [0.6]. Decision tree used the default settings. Random forest used numTrees: [15]. After reducing the number of rows, and performing preprocessing the final number of rows went from 2974335 to 2865076 and a total of 250 features that according to PCA covers 97.04% of the dataset.

Test	**State**	**LR**	**DT**	**RF**
1	all	56.5%	58.3%	53.9%
2	CA	65.5%	54.6%	54.7%
3	TX	70.7%	67.6%	60.8%
4	PA	75.4%	72.9%	64.5%
5	FL	69.8%	67.9%	53.3%

Table 6: Accuracy of each model based on 10 fold cross validation for each algorithm and on different states. Logistic regression (LR), Decision tree (DT), Random forest (RF) were tested.

5 Conclusion

The models performed quite poorly based on accuracy. However the the models perform better with individual states than running the entire dataset at once. This means that the time to operate is shortened and provide higher precision of the model. Based on the result from Spark the analysis, preprocessing and machine learning performed smooth and most of the operations were done in parallel. Since the classifiers had 4 values to choose from and data consist of none binary and non linear data which in this case make it more difficult to process a model. According to other models using the same dataset [5] that archived 95.4% accuracy for Logistic regression when using the original features without quantilization which means that it would be possible to improve the result but since the model crashed when using a big feature vector means that there are some limitations in hardware, which could be a cause of out of memory on the driver memory. Additional changes on the Spark parameter settings could be performed in order to run the model correctly. Future work may include refining and analyzing the dataset further, improving feature crossing to follow the theory and implementing an embedding for better performance are two steps that should be taken to obtain a value that can match reality.

[5] `https://www.kaggle.com/phip2014/ml-to-predict-accident-severity-pa-mont`

6 Discussion

One of the reasons that the model could have performed poorly is that rows that contain null are not discarded, instead of numerical values they are replaced with the median and for categorical values they get unique value as well in the one-hot-encoding. This is due to the fact that the amount of rows containing bad data (missing values/null) would account for almost the full set of data if not removed. In order to handle this some columns containing a lot of missing data is discarded and the rest is still applied. Replacing the original position features improved the efficiency of preprocessing, training time and reduced the feature vector. During preprocessing it noticeable that a feature crossing is not possible to do with one-hot-vector. This is due to that the vector size will be too big for limited hardware to handle and therefor take a lot of time to compute. Alternative to implementing one-hot or embedding, *Advanced analytics with spark: patterns for learning from data at scale* [17] suggests a method called FeatureHashing that specifies a given length and from there creates unique combinations of data instead of creating a unique feature for each unique class. Using featureHashing could potentially improve the clustering model in figure 5 and allow additional centroids to be placed to increase the accuracy. Additional limitations is on IBM cloud, since the service is not free and only a limited number of hours per month could be utilised (50) and python with spark used about 1.5 each hour means that the private docker service on the cloud is a must without upgrading the service plan.

7 Appendix

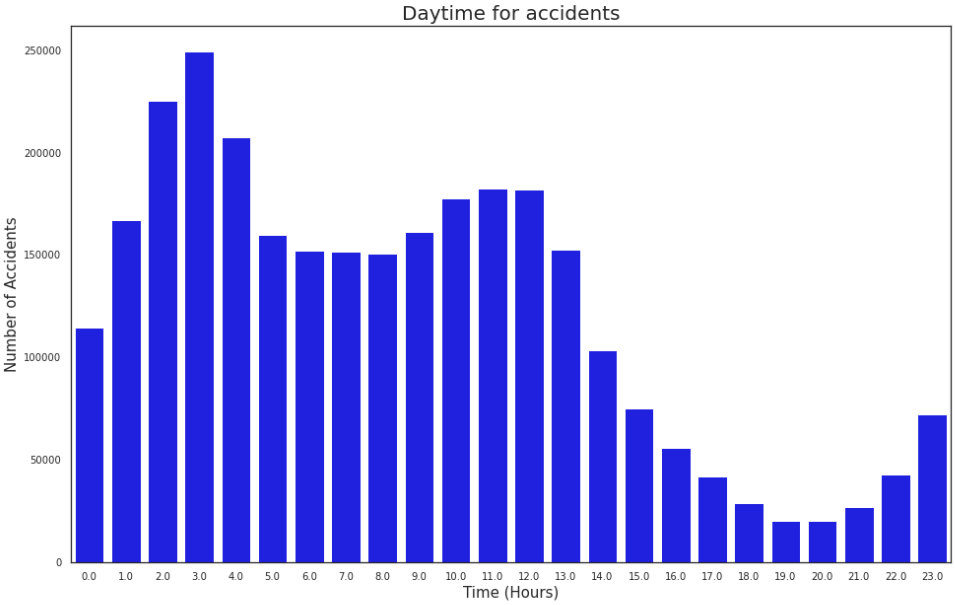

Figure 1: Which hour during the day that accidents occur.

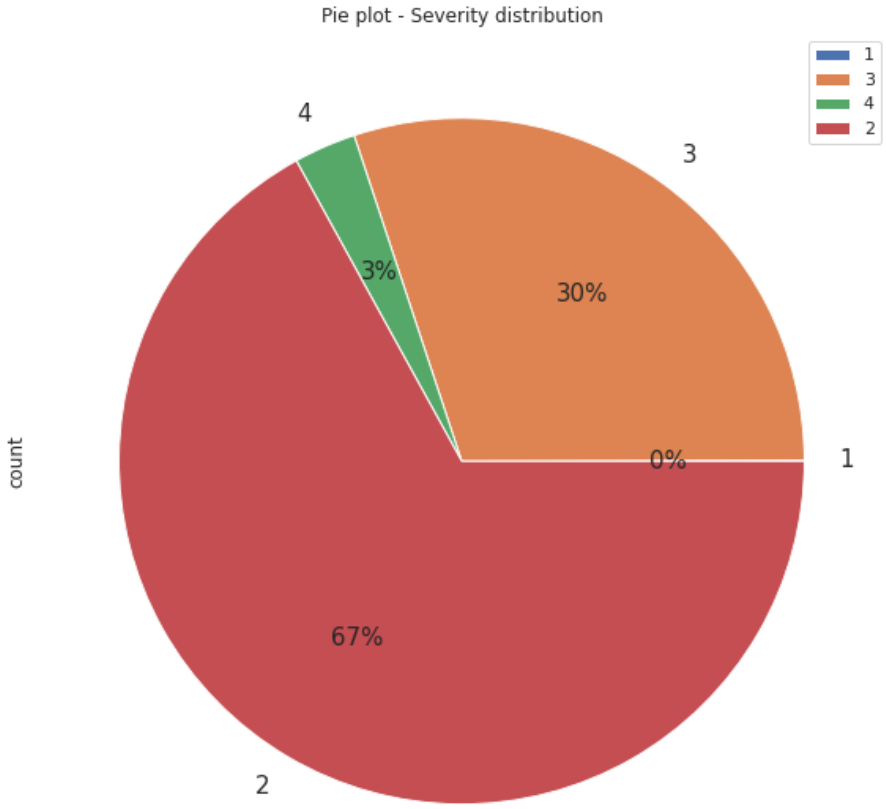

Figure 2: Distribution of severity.

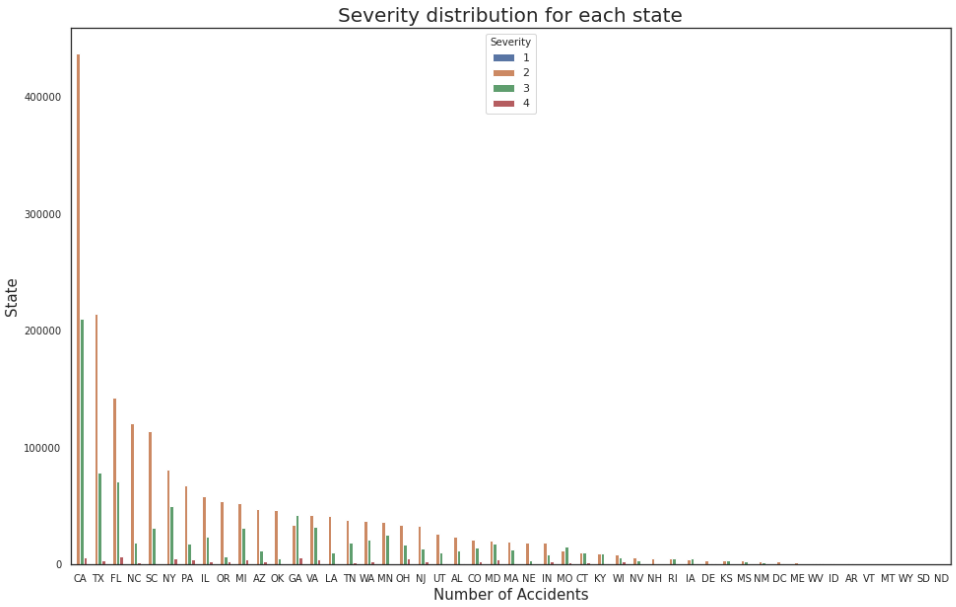

Figure 3: Distribution on severity in each state.

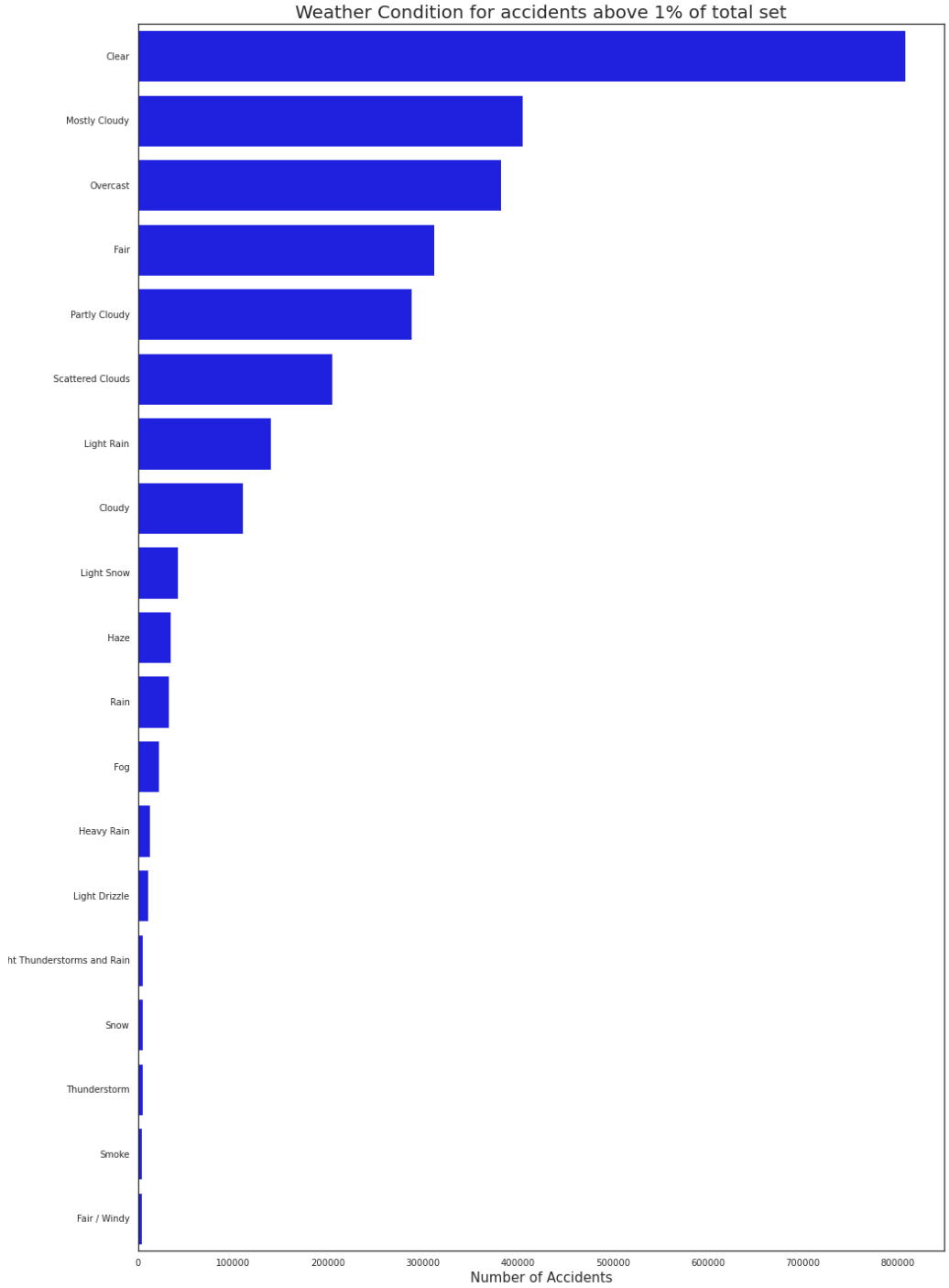

Figure 4: Top weather conditions for accidents.

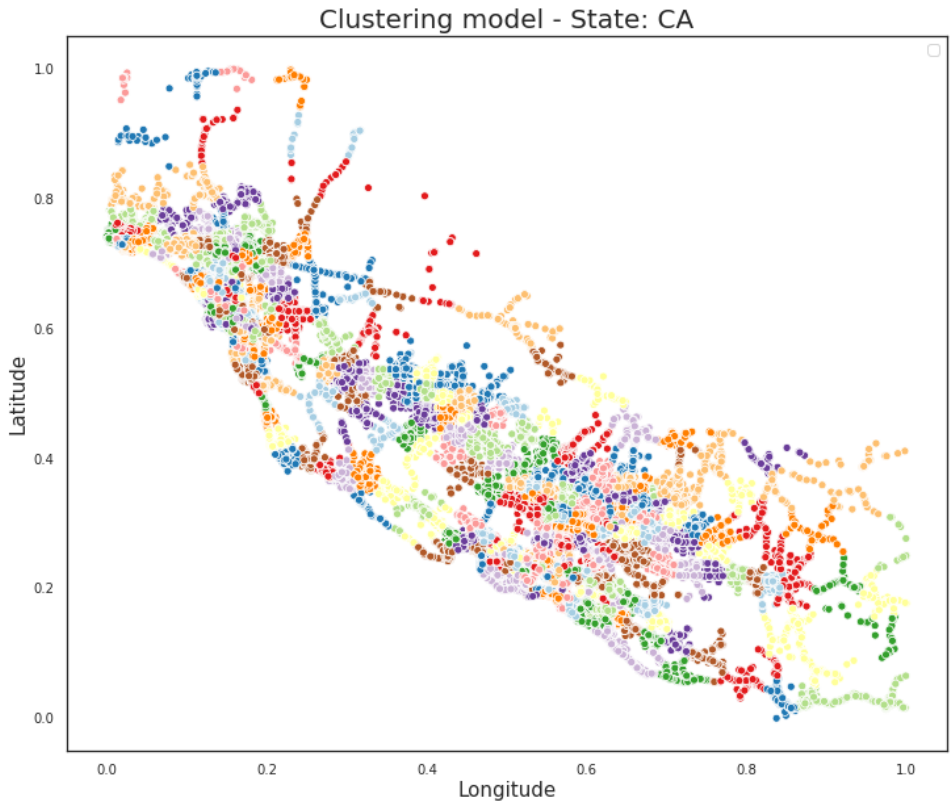

Figure 5: Distribution of centroids for the state CA, gives an overview of what the centroids are covering.

References

[1] Zhang Y, Cao T, Li S, Tian X, Yuan L, Jia H, et al. Parallel processing systems for big data: a survey. Proceedings of the IEEE. 2016;104(11):2114–2136.

[2] Leskovec J, Rajaraman A, Ullman JD. Mining of massive data sets. Cambridge university press; 2020.

[3] Hennessy JL, Patterson DA. Computer architecture: a quantitative approach. Elsevier; 2011.

[4] Moosavi S, Samavatian MH, Parthasarathy S, Ramnath R. A Countrywide Traffic Accident Dataset. arXiv preprint arXiv:190605409. 2019;.

[5] Moosavi S, Samavatian MH, Parthasarathy S, Teodorescu R, Ramnath R. Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems; 2019. p. 33–42.

[6] Hashem IAT, Yaqoob I, Anuar NB, Mokhtar S, Gani A, Khan SU. The rise of "big data" on cloud computing: Review and open research issues. Information systems. 2015;47:98–115.

[7] Ji C, Li Y, Qiu W, Awada U, Li K. Big data processing in cloud computing environments. In: 2012 12th international symposium on pervasive systems, algorithms and networks. IEEE; 2012. p. 17–23.

[8] Kossmann D, Kraska T, Loesing S. An evaluation of alternative architectures for transaction processing in the cloud. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data; 2010. p. 579–590.

[9] Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I, et al. Spark: Cluster computing with working sets. HotCloud. 2010;10(10-10):95.

[10] Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, et al. Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment. 2009;2(2):1626–1629.

[11] Ekanayake J, Li H, Zhang B, Gunarathne T, Bae SH, Qiu J, et al. Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM international symposium on high performance distributed computing; 2010. p. 810–818.

[12] Stupar A, Michel S, Schenkel R. Rankreduce-processing k-nearest neighbor queries on top of mapreduce. Large-Scale Distributed Systems for Information Retrieval. 2010;15.

[13] Wang K, Khan MMH. Performance prediction for apache spark platform. In: 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems. IEEE; 2015. p. 166–173.

[14] Yadav R. Apache Spark 2. x Cookbook. Packt Publishing Ltd; 2017.

[15] Wold S, Esbensen K, Geladi P. Principal component analysis. Chemometrics and intelligent laboratory systems. 1987;2(1-3):37–52.

[16] Liu H, Setiono R. Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence. IEEE; 1995. p. 388–391.

[17] Ryza S, Laserson U, Owen S, Wills J. Advanced analytics with spark: patterns for learning from data at scale. " O'Reilly Media, Inc."; 2017.