Spring 5-13-2024

Building a Data Pipeline and Machine Learning Model for Insurance Data

Connor Weyers
University of Nebraska-Lincoln

University of Nebraska-Lincoln

Undergraudate Thesis for University Honors Program

Building a Data Pipeline and Machine Learning Model for Insurance Data

Connor Weyers, BS
Computer Science
School of Computing

Supervised by
Prof. Vinodchandran Variyam

May 13, 2024

Contents

Abstract

Insurance telematics is an emerging and exciting field. It combines the advancements in GPS tracking, computational analytics, data processing, and machine learning into a useful tool to help insurance companies make the best product for their consumers. This is why National Indemnity looked to implement a telematics portion to their business processes of underwriting insurance policies and sponsored a School of Computing Senior Design project. In this report, we will first review existing solutions that been used to solve problems and subproblems similar to that we are given in this project. We then propose designs for the data pipeline and machine learning model that will optimal in providing predictions on the risk level of drivers. National Indemnity will be able to use this project to leverage predictions in order to optimize insurance rates to more accurately account for risk among the insured.

Keywords: Computer Science, Machine Learning, Insurance, Telematics, Data Processing, Data Analytics

1 Introduction

In the last twenty years there has been a fast development in the technology being used for all types of businesses. This had led to ever-increasing competition in the market to stay up to date with industry standards. The areas that have particularly revolutionized by technology include data gathering, data processing, and data analysis. Recently, machine learning had emerged as an important and popular data analysis method. A broad goal of machine learning is to algorithmically discover hidden trends in datasets. The improvements in machine learning models and computational efficiency saw machine learning being used in a variety of contexts. Machine learning began to be applied to the insurance industry with increasing frequency, including in this project for National Indemnity Company. But for machine learning to be best applied, it must have quality data. For this data, it needed to be gained through a source, usually through internal resources and with third party APIs. The data in this form must then be taken and put into some sort of data storage, so it can be used later. The data is then processed in some manner to ready it for the machine learning model which needs uniform data. The model would need to be trained and evaluated extensively to ensure that the model is accurate. The goal is to have the model gain insight from the data and make it easier for the internal insurance writers to write higher quality insurance.

1.1 Challenges for Machine Learning

Some problems for gaining relevant insights from insurance data is that it is constantly updating. The data must always be gathered to keep up with the large quantity of data in-flow. The data pipeline and machine learning model should be highly adaptable in order to keep pace with changing data structure and new technologies. Failure to do so would lead to performance diminishing as time goes on. As time goes on, new data sources will become available. These would be beneficial to incorporate making it crucial that the solution that is implemented is flexible to meet the needs of the highly dynamic business environment.

2 Research

In order to understand what would work best for our system, we first looked at the literature of similar projects to see how to best implement the highest quality system. We will take sources covering a variety of topics from data ingestion, data storage, data processing, and machine learning models. The majority of the sources used are studies discussing these features in relation to the insurance industry. There are also textbooks to give evidence to some of the more general features of a data processing and machine learning project. The insurance industry provides its own unique issues and circumstances for learning problems that are handled in a range of complex and unique solutions.

2.1 Data Ingestion

In machine learning, the data is the most crucial part. The data is generated by a variety of sources and must be ingested into a cohesive system where it can be further processed

and used later to train the model.

With insurance data, ingestion can be done in a variety of methods. These methods are determined by what kind of data the pipeline will be handling. The data coming in is normally through an outside source. The data is then received will be put into data storage for later use.

Data will also be gathered from several different sources. For insurance, these include but are not limited to telematics providers, mobile applications, other devices connected to the internet, and internal data. For ingestion purposes, each source will likely have its own format and security methods. Security risks on both sides of the connection should be managed with passwords, API keys, and other various security measures. Each source will need to be handled individually to ensure that the data is ingested properly.

Several variables should be taken into consideration and those are the data velocity, size, frequency, and format. These can vary considerably from source to source and a data ingestion plan should be able to handle these. The plan should also be robust enough to handle the expected changes of these metrics. The system should be able handle these metrics so that different formats, evolving and changing applications and data sources, data compression, and time constraints do not bottleneck any part of the process [MP18].

One thing that must be considered is whether the data pipeline will be ingesting data in streams or in batches. When the data is streamed in, it is processed as fast as it comes in. This works best in cases when the application is time-sensitive, and the data must arrive as soon as possible. In batch processing, the data is processed in larger volumes at set points in time usually determined by set intervals such as a minute, hour, etc. It requires fewer computing resources since it can be done offline. Batch processing works best when it is important that all the volume of the data is processed without urgent time constraints. Batch processing will also take additional intermediate storage in the time in between batches [BAL+20].

For most applications, it would be too expensive to produce a data ingestion pipeline completely from scratch. Many applications simply use a pre-built application that can be used to speed up development. Tools include DataFactory, Kafka, Flume, NiFi, Hadoop, and Sqoop among many others. This is much less time consuming than manually programming the ingestion and will usually fit the needs of the organization. Though each respective application has its own advantages depending on the ingestion method [MP18, ARGAT20].

2.2 Data Storage

Data in large quantities is stored by various means. To have an efficient system, the data storage mechanism must be well optimized for the data and for what the data will be used. These require careful tuning to build complex storage architectures. Common structures include data warehouses and data lakes. Additionally, there are additional architecture types such as NoSQL, SQL, and cloud storage.

Data warehouses are best practice when there is a large majority of structured data. This would be data that all comes in the same forms such as CSV or another rigidly structured data type. The data is typically stored in tables like formats and have strict relationships with other tables. This allows for higher speeds but lower flexibility. Data lakes are better when the data contains semi-structured or unstructured data. This would

be for file types that have little to no structure like JSON or text documents. Data lakes are best for these data types because they allow for more flexibility with data types. Structured data is also allowed in data lakes but would have less efficient access. Data lakes are best for handling raw and heterogeneous data [MPT20].

There is also the distinction between SQL and NoSQL data storage. NoSQL data solutions come in several different data models. These are key-value stores, columnar stores, document databases, and graph databases. Each of these data models is best suited for certain types of data and the application that will be accessing them. In general NoSQL uses fewer standard queries compared to relational databases seen with SQL databases. This makes the databases less able to port to other service providers and integrate with certain data types [MK19].

Cloud storage is another potential solution for data storage. Storing data on the cloud allows for easy and scalable storage for a company's data. It also allows efficient access from multiple locations at once. One downside is potential latency as the data is being accessed through an outside network. It is much more dependent on a larger external network. Additionally, entrusting data on an outside network with another company can be a security concern in some instances [KSF+20].

2.3 Data Processing

As data is first ingested into the system, it often comes in a raw format. Most machine learning models will not be able to handle this raw data. The raw data typically has a number of quality issues. Data can often be heterogeneous, inconsistent, and containing lots of noise. These issues vary greatly between datasets, so the solutions have a wide variance as well. There needs to be a number of steps in order to get this data ready for training and testing for the machine learning model.

The information in this section was sourced from the Data Mining textbook Data Mining: Know It All [Cea09].

The first step to transforming the data is to get it into the right format. This can be done efficiently with a number of algorithms to get the data into a proper format for the respective machine learning model that is going to be used. Incoming data is classified as structured, semi-structured, and unstructured data. When incoming data is unstructured, there is no standard method of converting this to model-usable data. This must be done on a case-by-case basis such as parsing through text files for a specific input. Machine learning models can often take in as input semi-structured data such as JSON or CSV with little to no overhead. This can be done as a temporary solution, but data performance is most efficient when it is converted into structured data that can go into a data warehouse. If incoming data is already in a structured format, then the only need would be to convert to a structured data type that best fits the specific implementation of the machine learning model.

Most data coming from real world sources will have some missing or incomplete data. This happens when there was some error in data collection, and it comes into the data pipeline as a null value or a zero when it does not make sense for there to be. There are several methods for handling these errors. The first and simplest method is to remove the entire instance of data. This is done to remove any instances that could cause incorrect results. This method can be problematic by removing too many instances since it has the

potential to drastically reduce the dataset. The other method is to replace this missing value with some sort of average value. This can be the mean, median, or mode of the entire dataset or a subsection of it. This method is helpful because it will retain average values for each field. One issue with this method is it could bias the results to favor the more heavily favor average cases resulting in biased results. The last common method is to copy the value of the instance above or below. This method will keep the randomness but could bias the results since it could result in uncommon values being duplicated more. Another method would be to remove the entire column. This method would be good only in cases where there are many missing values in the field. In this scenario, all other methods would result in biased results so it can be concluded that the column cannot add any value to the model.

For a lot of insurance data, there will likely be several data types including strings and integers. Machine learning models cannot process strings on their own so they must be transformed into integers. The strings in these scenarios are usually representative of certain categories. These categories can be encoded as numbers. The simplest method of encoding these would be to assign an integer value to each category and give each string value to its corresponding integer value. This is referred to as ordinal encoding. This method retains the number of fields which helps keep down computation. One issue with ordinal encoding is that machine learning models will treat the categories as numbers so they will essentially be ranked. If the data type is not in a ranked order, this method should not be used. The next method is one-hot encoding. In this method, each category is given its own field and will be given a 0 or 1. This method provides better accuracy when considering categorical fields. The issue is this could bloat computation with more fields. Since only one of these new fields will have a nonzero value, there is going to be a lot of excess computation, especially in cases with many fields. Another common method used for encoding is called target encoding. This works by calculating the average value of the target variable based on each category in the dataset. This average value is used instead of the string. This method works well by not inflating the number of fields while also adding information on how the categories relate to one another. The drawback of this method is that the averaging could introduce extra bias into the dataset [sci].

For large data models, there is often extraneous and duplicate data. Keeping all this data in can lead to a bloated and slow model. One key step to inputting data is to reduce the size of the data. This will boost computation speed but could lead to less accurate results. The key to reducing data is to find a method that extracts relevant data while discarding extraneous data. There are three broad categories to reduce data size. They are dimensionality reduction, numerosity reduction, and data compression. Dimensionality reduction is done to reduce the number of fields input to the model to reduce the computational burden of the model. The most common methods are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Generalized Discriminant Analysis (GDA). PCA uses orthogonal mapping to map a large dimension dataset to a lower dimension set. It is good to be used for mapping relationships between variables. The goal of PCA is maximize variance across the dataset. The next most common method is LDA. It works by projecting a dataset onto a lower set of variables. It is quite similar to PCA but has a goal of maximizing differences between classes. This allows the ML model to more easily determine classes. In numerosity reduction, the data is represented in smaller forms to reduce the volume of the data. Numerosity reduction

methods are split into parametric and non-parametric methods. Parametric methods replace the actual data with parameters using a model. Non-parametric methods do not use parameters and include histograms and clustering. Data compression uses algorithms to apply transformations to the data so that it takes less storage. Compression techniques are also split into two categories. The first is referred to as lossless and is when the transformations can be reversed with no information lost. It is referred to as lossy when the data reduction loses some information in the process.

Another important step to preprocess large data is to transform it. This can be done with methods such as smoothing, attribute construction, aggregation, and normalization. Smoothing is done with a variety of methods to reduce noise in the data in order to better estimate the trends of the data. Attribute construction is done to add new features are added to the dataset to extract more information from the data. Normalization is the process of converting the data to be within a smaller numerical range. This is done with a variety of techniques to standardize the data so that it is weighted properly.

2.4 Machine Learning Models

There are several types of machine learning models that are prevalent in different areas of industry and research. The next subsection gives a brief overview of some of the most common machine learning models. The section following gives an analysis of model performance in the insurance industry.

Model Overviews

The majority of the content in this section is sourced from the Introduction to Machine Learning textbook by Ethem Alpaydin [Alp20].

One method is to use a clustering model to group the client data into groups. Clustering is an unsupervised machine learning technique that does not need labeled data to make inferences. After the clustering, there are a number of different models to be used. There can be decision trees, neural networks, or others. The advantage of this method is that the results are better in clusters. Some disadvantages are multiple distinct stages that will take more computing power to train and run as well as harder to tune the hyperparameters of multiple models. This method will best work for data that has more distinctive clustered groups. This method is our proposed method of operation for our data. We would like to identify different risk categories that can be grouped into clusters. The extra method on top of that would be to predict a risk score or an estimated insurance premium.

Clustering can be done in a multitude of ways too. K-means clustering is a method that is tried and true but would take a lot of tweaking to get the right number of clusters for optimal performance. Like any machine learning algorithm, there are hyperparameters that need to be manually adjusted and we must use the literature and trial and error methods in order to find what best works for our individual project. Fuzzy clustering is a method that builds off traditional k-means clustering to have instances that can be in multiple clusters.

Clustering can also be used as part of the preprocessing process to reduce or transform the data. Clustering is an immensely helpful and versatile tool that can be used in a

variety of use cases for machine learning.

After clustering or without clustering, there are numerous methods to use. They all include some level of supervised learning. Some common methods are using decision trees, random forests, boosting, support vector machines, neural networks, and k-nearest neighbors.

One of the simplest machine learning models is the decision tree. This works by training the model to divide a dataset in a hierarchical manner until a subset of the data is uniform enough to make an accurate prediction. The simplicity of this model allows it to be more easily understood. It is very efficient to train. The simplicity of decision trees does have a drawback in that it can only handle relatively simple problems. It is often prone to overfitting on training data and thus having inaccurate results when applied to other data. There are a number of ways to reduce overfitting in individual decision trees but those require more complex implementations.

Ensemble methods are powerful techniques for enhancing predictive performance by combining multiple simpler models. One category of ensemble methods are random forest models. These work by using many decision trees. They are able to harness the simplicity of decision trees and scale it to larger and more complex data. The training set is split up between the models so that each tree is trained on a different subset of the data. When the model is run, it obtains a prediction from each individual tree. These predictions are aggregated and averaged to give a single prediction. When splitting data between the trees, the randomness is increased. This makes it so that the overall prediction avoids overfitting and biased data.

Boosting, a popular ensemble method, sequentially trains a series of weak learners, each focusing on the instances misclassified by its predecessors. Through this iterative process, boosting aims to improve the overall predictive accuracy of the model. One well-known ensemble method is the random forest algorithm. Random forest constructs a multitude of decision trees during training, where each tree is trained on a random subset of the data and features. The final prediction is then determined by aggregating the predictions of all individual trees, typically through voting (for classification) or averaging (for regression). Ensemble methods like boosting and random forest often outperform individual models by leveraging the strengths of multiple base learners and mitigating their weaknesses. They are robust to overfitting and noise. Due to this, they can handle high-dimensional data effectively. However, ensemble methods may require more computational resources and parameter tuning compared to single models. Despite these considerations, boosting and random forest are widely used in practice due to their versatility and ability to deliver high-quality predictions across various domains and datasets.

Support Vector Machines (SVMs) are another method of machine learning that is used extensively in the insurance industry. They work by mapping out the data points in an n-dimensional space. The goal of classification is to find a plane that can separate the points of classes with as much distance between the classes as possible. This can be done in n-dimension, but more accurate results are procured in higher dimensions. Modern SVM models work by applying kernels to transform the data to a higher dimensional space so that the classes are more easily separable. SVMs can produce very accurate results but can also be computationally intensive. They are particularly useful in cases where the dataset has a high number of dimensions. SVMs also need added complexity

when dealing with regression or classification problems with more than two classes.

Neural networks represent a class of machine learning models inspired by the structure and function of biological neural networks in the human brain. These models consist of interconnected layers of artificial neurons, organized in a hierarchical fashion. Each neuron receives input signals, processes them through an activation function, and generates an output signal that is passed to the neurons in the next layer. Neural networks are capable of learning complex patterns and relationships in data through a process called training, where the model adjusts its parameters iteratively to minimize the difference between its predictions and the actual targets. This training process involves forward propagation to compute predictions, followed by backward propagation of errors to update the model's parameters using optimization algorithms like gradient descent. The most widely used neural network architecture is the feedforward neural network, where information flows in one direction, from input to output layers. Additionally, deep neural networks, with multiple hidden layers, have gained popularity for their ability to learn intricate features from high-dimensional data. Neural networks have demonstrated remarkable success in various domains. However, they also have a high computational complexity and consequentially need substantial amounts of computational resources. These models also require large amounts of labeled training data to have accurate results. Results can also drastically vary depending on many different hyperparameters and model architecture. Despite these challenges, neural networks are a state-of-the-art technology that can produce extremely accurate results.

One popular method for making predictions based on existing data is the k-nearest neighbors (KNN) algorithm. KNN is a simple, yet effective supervised machine learning technique used for classification and regression tasks. In KNN, each data point is classified based on the majority class of its k nearest neighbors in the feature space. The "k" in KNN represents the number of nearest neighbors considered for classification, and it is a hyperparameter that needs to be specified by the user. The algorithm calculates the distance between the query instance and all the training samples to find the k nearest neighbors. Once the neighbors are identified, the algorithm assigns the most common class label (for classification) or the average of the values (for regression) among these neighbors to the query instance. One advantage of KNN is its simplicity and ease of implementation. Additionally, KNN can be robust to noisy data and can handle non-linear decision boundaries. One drawback is that this model may perform worse in cases of high dimensionality, particularly when the number of features is large. Another issue is that the choice of the value of "k" and the distance metric used can significantly impact the performance of the algorithm as do many hyperparameters in other models. Therefore, careful parameter tuning is essential for optimal results. Despite its limitations, KNN can be particularly useful for datasets with well-defined clusters or when the decision boundary is complex and difficult to capture with other algorithms.

In the Insurance Industry

In a 2016 study, researchers sought to find the best model to predict the profitability of insuring customers. The researchers used a metric called the customer lifelong profitability computation model to determine the profitability of customers. They used random forest, a generalized boosting model, linear regression, a support vector machine, and a decision

tree. Using the Root Mean Squared Error they measured performance. They found that the order of performance from highest to lowest performing was random forest, generalized boosting machine, support vector machine, decision tree, and linear regression. [FJS16]

In a 2019 study, researchers used XGBoost and logistic regression to predict claims occurrences. The dataset used in the study was gleaned from telematics data that contained information on driving behaviors. The dataset also contained information on what claims the drivers had made. Upon obtaining results, they found that XGBoost performed far better on accuracy, specificity, and sensitivity [PNGA19].

In a 2020 study, researchers tried Naive Bayes, neural network, J48 (a decision tree algorithm), and XGBoost models to predict the likelihood that an insured driver will submit a claim. They used online data from Kaggle. They used various preprocessing techniques like imputation, discretization, encoding, and standardization before the data was ran on the models. They found that XGBoost and J48 performed the best on the given task. They had the highest accuracies and were the quickest to train [AEA20].

In a 2021 study, it was investigated which machine learning model would work best on predicting claims insurance. The study uses data from Porto Seguro, a major Brazilian insurance firm, aiming to develop a machine learning model for assessing driver risk. Emphasizing the importance of model generalization for reliable predictions, the study addresses binary classification scenarios where claims are categorized as either 1 (will file a claim) or 0 (will not file a claim). By exploring various classification algorithms, the research aims to identify the most effective approach for accurate claim prediction, aiding insurance companies in risk assessment and policy customization. They looked at various models including random forest, several decision tree algorithms, XGBoost, KNN, and Linear Regression. They found random forest, C50 (a decision tree algorithm), XGBoost, and J48 to be the best performing. They evaluated using accuracy, error rate, kappa, area under the receiver operating characteristic curve (AUC), sensitivity, specificity, precision, recall, and F1. Their RF model performed the best on every metric except for specificity. C50 was the next best performing on most metrics. XGBoost and J48 consistently were the next best models [HM21].

In a 2023 study, a team looked into forecasting auto insurance claims. The dataset they use is from various sources in Athens, Greece over a period of 12 years. In addition to claims data, they used multiple independent variables including weather data and car sales to predict the mean motor insurance cost. They used support vector machines, random forests, and XGBoost to predict this value. They determined the fifteen most relevant variables and trained the models with only those fifteen and another time with all the variables. They found the models performed best with only the fifteen most relevant variables. They found that the random forest model with a limited depth performed the best, with XGBoost achieving the second highest performance [PGPZ23].

3 Data Pipeline Implementation

This section will give a description of the implementation used in this project to ingest and prepare the data for use in the machine learning model.

3.1 Data Ingestion

This project required large amounts of data from third party providers. Agreements were made with the companies and relevant credentials were given to be used in the project. Since the data was coming from different providers, different data ingestion pipelines will be used to get the data into data storage. For data storage, see section 3.2 for more information on implementation.

There are generally two categories of data ingestion implementations. The first is to stream in the data. This means that data is ingested in real time from the provider. This is done when a project is time-sensitive and a delay in input would have a significant negative impact on the project. The second category is to ingest the data in batches. This method does not ingest data as soon as it is ready, rather it processes data at set intervals.

This project was better suited to ingesting data in batches. The machine learning model was not trained in real-time. It did not need the data as soon as possible. Additionally, the amount of data ingested from the providers would be more than what would be cost-effective for a streaming method. Streaming adds a much higher degree of complexity to the implementation. Batch ingestion allows for relatively standard processing times, data size, and frequency. This allowed for an optimal ingestion method for this project.

For this project, the data used had already been gathered without a set ingestion scheme. It will be up to future projects to fully realize the implementation of the ingestion.

Since the project did not require any complex ingestion method, the common applications for data ingestion more than sufficed. Azure DataFactory was the best for this project as it allowed for seamless integration to other Azure products that were used in the later parts of the pipeline.

3.2 Data Storage

For companies like National Indemnity that are not in the tech industry, it is common to use third parties for data storage as it is extremely expensive to establish their own servers and operating systems. National Indemnity has the preference of Azure and Microsoft services so that is the primary software lender the project used.

For insurance data for our machine learning system, a data lake was the best option. The data consisted of text documents, tables, and third-party telematics data in JSON format. The data was also coming into our system raw and unprocessed, so our system needed to be able to handle the storage of all the stages of the data in the pipeline. A data lake was the best option for our project because data lakes can handle the input data is given, as well as handling the data in intermediary processing steps. The data lake would also be able to store the testing results without being constrained by a strict data format.

Since the project did not have fully integrated ingestion, the data storage was also not fully implemented. This will also be done by future projects to ensure efficient use of data within the project. The best option for this would be using Azure Data Lake Storage as it would provide the necessary resources to build an efficient data lake that can be integrated well with the data ingestion and other systems operated by National Indemnity.

3.3 Data Processing

One of the biggest issues with machine learning models is not the models themselves but the data that they need to function. Most data collected in the real world will not fit the format necessary to give the models quality results. The data in this project was collected primarily by third party sources with telematics technology. This data is given to the system in JSON format and will be stored in the data ingestion phase.

Once the raw data is captured, the data must be transformed into usable data that can be used to train the ML model with. This involves the processes of imputing, text processing, grouping, outlier detection, event monitoring, feature engineering, data splitting, normalization, dimensionality reduction, duplicates, noise reduction, and handling time series data.

The data for the projects came from various sources so there are differences between the formats. The first step of the processing was to get the data organized and in a format that can be efficiently modified and analyzed in later steps. There are three main types of data. There is summary trip data, telematics data, and claims data. All these kinds of data may be separated based on the provider of the data. The files would be read and joined using the Pandas library to create new JSON files with all the relevant data. These files are saved to the database for later processing.

The next step was imputing data. There are several different methods of imputation. There are a variety of methods of dealing with missing data. Many of the rows may have missing values in one column due to the different service providers reporting distinct kinds of data. The best course of action in this scenario was to drop these columns from the data since filling in these values could result in extreme bias when there is the number of rows is so significant. Some columns were crucial to later processing and the model. These include fields like the vehicle identifier. This information could not be inferred without inducing noise in the results, so these rows were dropped. For less crucial missing values, the field was imputed by calculating the average between thirty neighboring rows. This number was chosen to reduce randomness in the imputed value but also low enough to still be able to compute it quickly.

Another action taken in the preprocessing stage was aggregation. This is the process of taking a summary of a certain portion of the data. The project did this on the telematics data. This data had a multitude of rows since it was collected in real time with coordinates, speed, and direction among other fields at short intervals. There were a number of these rows for each vehicle's trip. We aggregated this data by finding several different metrics that can be pulled from this telematics data. This process could also be considered to be feature engineering. Speeding events were used from the telematics data. This was done by comparing the speed to the speed limit. If a driver is over the speed limit excessively or for an extended period of time, they will have reported speeding events. Additionally, harsh acceleration and deceleration events were captured and reported. This was to determine if a driver brakes hard or speeds up quickly. These trends are pulled from this data so the model will have more to analyze. These features were pulled from the telematics data to accurately depict the information described so the model can better understand the trends in the data.

In addition to the features engineered from the telematics data, there was additional feature engineering being done. Processing was done to determine the percentage of time

a driver drives at night compared to during the day which would likely affect the average risk since there would be lower visibility. Several other features were used by connecting with open-source APIs. It was determined where a driver is likely to stop for extended periods of time which is recorded as a garaging location. This garaging location is then inputted into these APIs which will determine the amount of vehicular crime and natural disasters in that area. These were used to give the model data that could be used to infer risk.

After the features were created, the data then needed to be normalized. This involved making the integer values fit in a set range of values. For this project, several columns were normalized to a range of 0 and 1 on a linear scaling. This was done so the range of values are standard across fields. This prevented bias between fields with different ranges of values.

Several fields in the data were strings in their raw form. Outside of some natural language processing models, these fields need to be converted into numerical data before being inputted into the model. For several applicable fields, we directly converted the field to an integer value that corresponds with the ranking of the string using ordinal encoding. Since our dataset did not have many non-numerical fields that could not be directly converted to a single integer, we found it best to use one-hot encoding. This preserved the semantic value of the field. Due to a lower number of fields that this was done to and a small number of options per field, the computational slowdown due to increased dimensions is limited.

Once all of the steps of the preprocessing are finished, the data is ready to be used in the model.

4 Machine Learning Model Implementation

Machine learning is a broad field with a variety of different models to choose from. Each model has its own unique benefits and challenges. Every problem in the real world contains its own unique circumstances that must be carefully considered. The model must be chosen to best fit the problem. Additionally, there is a multitude of further adaptations and hyperparameters to choose from that make the decision extremely hard to get right.

The goal of this project was to determine the risk of a given driver. The dataset used in the project contained speed, direction, time, and other telematics data. After processing, the data is ready to be fed to a model. Since the project was looking to predict a data point, that narrows down the models to supervised models. Purely unsupervised models like clustering were ruled out. The most commonly used models for similar problems are decision trees, random forests, boosting, support vector machines, and neural networks.

Though a purely clustering model would not be useful for the current project, using a clustering model and then a supervised model on top would be possible. This option would first group the dataset into related groups. Then a supervised model could be trained on each of the clusters individually, providing more tailored results on each cluster. The issues with this approach would be that it could result in overfitting since each model instance would only be trained on a small subsection of the data. This method could also be computationally expensive as well as disjointed by breaking up training into

unsupervised and supervised on multiple clusters. This approach was deemed to be not optimal for this project.

Support Vector Machines are efficient models that can be used in a variety of applications in the insurance industry. They are typically highly accurate, especially in high dimensional spaces. The dataset used in this project had a number of dimensions but does not have such a high amount that support vector machines would be able to provide a relative advantage to other models. Support vector machines can also be expensive, especially on multiclass problems. The project will predict a risk score which would need additional complexity in a support vector machine. A support vector machine would not be the best choice for a problem like this.

Neural networks are good for the project because they are typically highly accurate. The issues with them are that they are typically costly in terms of time and energy. The project does not need to train the model frequently but keeping cost low is always a high priority. Additionally, neural networks are hard to understand. They work as a black box that takes in inputs and does a number of intermediary steps before outputting a prediction. While the input and output layers are well defined and easy to understand, everything that happens in between is largely incomprehensible for a human. This means neural networks are hard to fine tune if the model is not performing up to standards. This lack of understanding also poses a problem to the business aspect since it is hard to explain to non-technical users what the model is doing.

Decision trees have the advantage of being simple and easy to train. The simplicity of these models makes them impractical for a high dimensional dataset and complex problem like the one in this project.

Though singular decision trees are impractical for use in this project, the random forest model has been used in many different applications in the insurance industry. Several models like J48 and C50 are the most common. They work by producing multiple decision trees and training each one on subsets of the overall dataset. This provides randomness across the models which reduces overfitting and increases accuracy. Random forest models provide some of the highest accuracies in many machine learning studies in the insurance sector. Random forests maintain the simplicity and efficiency of training individual trees. This makes random forests one of the most adaptable and widespread models in use. Training a random forest model can take a long time unless it can take advantage of parallel computing resources.

Gradient boosting models are some of the most commonly used methods in the insurance industry. XGBoost stands out as the most popular among these models. It was used in many recent studies involving similar problems to the one in this project. XGBoost and other ensemble methods were usually at or near the top in terms of performance. It is highly accurate, and due to the nature of its training, is faster than most other models. The nature of its training also allows it to be more easily understood. It can be more finely tuned to suit the needs of the project by determining what parts of the training to focus on. The combination of the quickness and versatility of this model allows potential non-technical members of the company to be able to use the model to suit their needs based on updating driver habits, and changes in the business.

We found that gradient boosting and random forests to be the best choices to develop a model on. In studies that covered machine learning models on similar issues as in this project, the top performing models were these two models. These performed the best

in most categories measuring accuracy. They were the best fit to handle the complex dataset of this project while also providing efficient training times.

We chose XGBoost model because it can provide the fastest training times using boosting techniques that allow it to better learn the trends in the data. The project will also be modified frequently in the future by non-technical users at the company, which made interpretability to be a principal factor in our decision making. We found that XGBoost would allow users the most intuitive way of adjusting parameters to train the model how they best see fit. It would allow them to specify what parts of the dataset the model should focus its training efforts on so that the users who are experts in their respective field can apply their knowledge without dealing with complex machine learning concepts as might be the case with tuning other models.

The XGBoost model that has been created takes in the processed data from previous steps and trains on various features to be able to get the best results. It can accurately predict a driver risk score. The model can be tuned so that it focuses more on features like location, speeding, or braking that would need to be changed depending on what the user finds valuable given the context of the business and other environmental factors.

The project consisted of a graphical user interface that could be used by those at National Indemnity to train the model. The interface allowed the user to choose the data that the model will train on. Additionally, the user could change the parameters to the model to finely tune the model to the specific dataset of the user. The flexibility and interpretability of XGBoost allows the interface to take advantage of this to make it so that non-technical users can adequately harness the power of the model.

5 Conclusion

In this paper, we outlined the implementation of an effective data pipeline and machine learning model for assessing driver risk in the auto insurance industry. Through efficient data ingestion, storage, and processing stages, we ensured that raw data from a multitude of sources was able to be transformed into a usable format for training our machine learning model.

We prepared a rich dataset capable of feeding into our machine learning model effectively. To do this we utilized a data lake for storage and employed processing techniques including imputation, aggregation, feature engineering, and normalization,

In selecting a suitable machine learning model, we evaluated various options including clustering, support vector machines, neural networks, gradient boosting machines, decision trees, and random forests. Ultimately, we determined that gradient boosting and random forests, with XGBoost as our preferred implementation, offered the optimal combination of accuracy, efficiency, and interpretability for our project's needs.

The XGBoost model developed in this study not only demonstrates accuracy in assessing driver risk but also provides a user-friendly interface for non-technical stakeholders to fine-tune parameters based on their domain expertise. This adaptability ensures that the model remains relevant and effective amid evolving business requirements and environmental factors for the company.

By integrating robust data pipeline implementation with a sophisticated machine learning model, this project lays a foundation for enhancing risk assessment strategies

in the insurance industry, contributing to improved decision-making and operational efficiency within the field.

Bibliography

[AEA20] Shady Abdelhadi, Khaled Elbahnasy, and Mohamed Abdelsalam. A proposed model to predict auto insurance claims using machine learning techniques. *Journal of Theoretical and Applied Information Technology*, 98(22):3428–3437, 2020.

[Alp20] Ethem Alpayd. *Introduction to Machine Learning*. MIT Press, Cambridge, MA, 2020.

[ARGAT20] Jaber Alwidian, Sana Rahman, Maram Gnaim, and Fatima Al-Taharwah. Big data ingestion and preparation tools. *Modern Applied Science*, 14:12, 08 2020.

[BAL+20] Sarah Benjelloun, Mohamed El Mehdi El Aissi, Yassine Loukili, Younes Lakhrissi, Safae Elhaj Ben Ali, Hiba Chougrad, and Abdessamad El Boushaki. Big data processing: Batch-based processing and stream-based processing. In *2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, pages 1–6, 2020.

[Cea09] S. Chakrabarti and et al. *Data Mining: Know It All*. Morgan Kaufmann, Burlington, Massachusetts, 2009.

[FJS16] Kuangnan Fang, Yefei Jiang, and Malin Song. Customer profitability forecasting using big data analytics: A case study of the insurance industry. *Computers Industrial Engineering*, 101:554–564, 2016.

[HM21] Mohamed Hanafy and Ruixing Ming. Machine learning approaches for auto insurance big data. *Risks*, 9(2), 2021.

[KSF+20] Christian Kaiser, Alexander Stocker, Andreas Festl, Marija Djokic-Petrovic, Efi Papatheocharous, Anders Wallberg, Gonzalo Ezquerro, Jordi Ortigosa Orbe, Tom Szilagyi, and Michael Fellmann. A vehicle telematics service for driving style detection: Implementation and privacy challenges. In *VEHITS*, pages 29–36, 2020.

[MK19] Andreas Meier and Michael Kaufmann. *SQL & NoSQL databases*. Springer, 2019.

[MP18] Andreea Mătăcuţă and Cătălina Popa. Big data analytics: Analysis of features and performance of big data ingestion tools. *Informatica Economica*, 22(2), 2018.

[MPT20] Bálint Molnár, Galena Pisoni, and Adam Tarcsi. Data lakes for insurance industry: Exploring challenges and opportunities for customer behaviour analytics, risk assessment, and industry adoption. 07 2020.

[PGPZ23] Thomas Poufinas, Periklis Gogas, Theophilos Papadimitriou, and Emmanouil Zaganidis. Machine learning in forecasting motor insurance claims. *Risks*, 11(9), 2023.

[PNGA19] Jessica Pesantez-Narvaez, Montserrat Guillen, and Manuela Alcañiz. Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, 7(2), 2019.

[sci] scikit-learn Contributors. Preprocessing data. https://scikit-learn.org/stable/modules/preprocessing.html. Accessed: May 6, 2024.