University of Groningen

Scalable Computing

Appliance Lifetime Prediction

GROUP - 25

Authors

Anil Mathew	S4056167
Swastik	
Siddharth Baskaran	S3922782

April 17, 2020

Contents

1 Motivation

Is it possible to know how many years the household appliance will endure and function properly before wear and tear? To effectively manage the household budget, it is critical to know the average life expectancy of the appliance used in the house. All the questions looming about appliance life expectancy by the customers can be put to rest with the help of our solution that uses the power of big data to predict the possible life expectancy of the appliance based on various parameters.

With this solution, the customers are provided with useful insights on the best way to help appliance age well and have a long, healthy life.

As part of the course Scalable Computing, the team was asked to create an application that would assist the users to understand how many years the appliances available in their household might still run or if it has outlasted our training data.

2 Dataset

The team did not find a real-world dataset for training that contains the required information in order to process the prediction for the application. We, however, got a dataset that contains a few interesting columns from `https://www.kaggle.com/taranvee/smart-home-dataset-with-we` and then applied ETL to get the desired outcome. As the core idea of this prediction is based on the energy consumed and age of the appliance. We extracted and transformed the raw data using the below steps and this is represented in Figure 1:

- Energy consumed(kW) for appliances: Dishwasher, Furnace, Fridge, Wine cellar, Garage door, Microwave from the mentioned dataset from kaggle. We have restricted the appliances to 6 for this demo.
- For the age of the appliance, we simulated by adding two attributes: Purchase_date and Capture_date (applianceAge =Capture_date - Purchase_date).

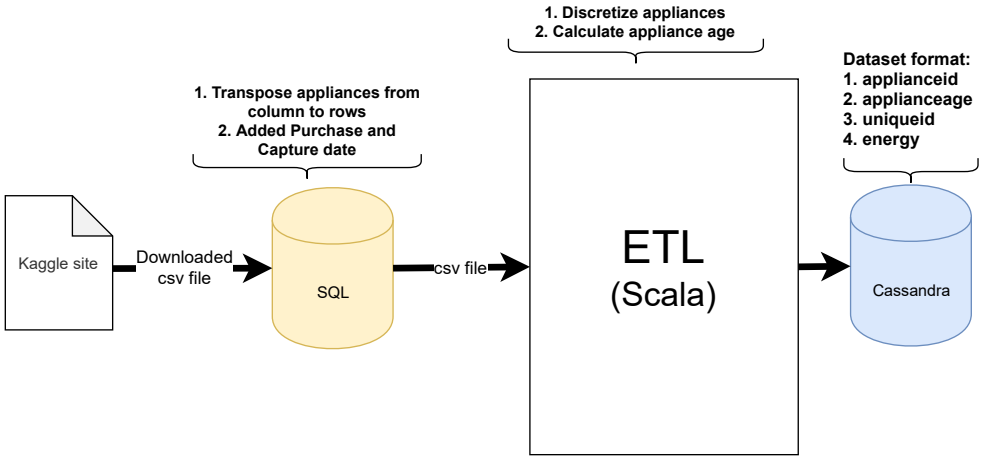

Figure 1: Dataset ETL process

3 Technology/Infrastructure

The below are the technologies and the infrastructure that the team has used for this project:

- **Scala** : Spark has been developed using Scala. Hence, the team has decided that it is the most apt decision to select it as our core programming language to develop the Data generator and Spark application for appliance lifetime prediction. The advantages of using Scala is because it is scalable, concise, strongly typed, immutable-first language, non-existent boilerplate codes, interoperability with java, easy to maintain, leverage the advantages of JVM.

- **Spark** : Apache Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. We will be using Spark to handle big data and make use of the power to parallelize the algorithms.

- **Kafka** : We have used Apache Kafka for building real-time streaming data pipelines that reliably get data between systems or applications. The stream/batch data is provided with the capability of high throughput, lower latency, fault-tolerant, and durability with the use of this technology.

- **Cassandra** : Apache Cassandra database is used for scalability and high availability without compromising performance. It is a perfect fit for consuming lots of time-series data that comes directly from users, devices, sensors, and IoT devices.

- **Docker** : Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. We use Docker to containerize our spark applications that can then be used by the Spark Operator to use it in Kubernetes.

- **Kubernetes** : Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications. We have used this container orchestration engine to handle the fault tolerance and scalability of the application. Spark, Cassandra, and Kafka components have been deployed on Kubernetes.

- **Tableau** : Tableau is a powerful and fastest-growing data visualization tool used in the Business Intelligence Industry. Data analysis is very fast with Tableau, and the visualizations created are in the form of dashboards and worksheets. With this BI visualization technology, the end users can view the predicted lifetime result for the appliances and also interesting graphs to understand the trend in the data.

- **Google Kubernetes Engine(GKE)** : This is a management and orchestration system for Docker container and container clusters that run within Google's public cloud services. Google Kubernetes Engine is based on Kubernetes, Google's open source container management system. The team has used GKE due to its simplicity and free credits.

- **SBT** : This is the selected build tool to handle continuous integration, profiling, and dependencies across the project.

4 Architecture

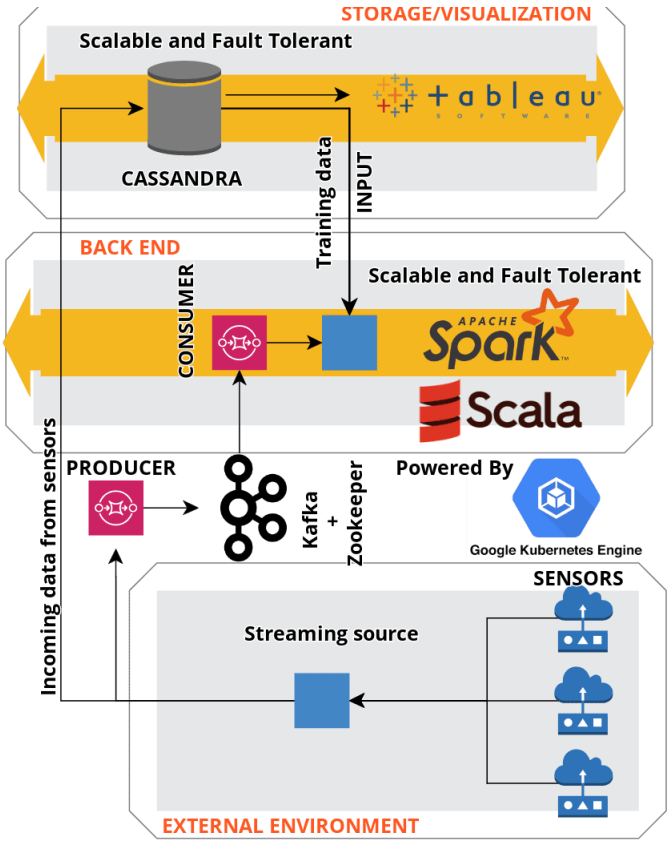

Figure 2: Architecture of the application

From Figure 2 we can observe the architecture that we have decided to use for this project. The key components include **Apache Kafka**, **Apache Spark**, **Scala**, **Cassandra**, **Tableau** and **Google Cloud Platform**.

As per the architecture diagram we can infer that the input data comes from various sensors that are present in the appliances. We collect this data through a streaming source and send it to either Cassandra for batch processing or to Kafka as a streaming source. In our case we use a sample data set to simulate the data obtained from these sensors. Next, in Kafka we have 2 topics, *Streaming Topic* and *Batch Topic* to process this data obtained from the sensors, as spark natively supports streaming, we then send the streaming topics data to Spark streaming and similarly the batch data is also sent to Spark, where we then run either KNN or K-Means algorithm to predict the life expectancy of each appliance, and the computed results are then stored in Cassandra under the *prediction_kmean* or *prediction_knn* table which is then used by Tableau to help visualize the data.

This entire setup is configured on Docker and Kubernetes so as to make it scalable and fault tolerant and eventually, this whole system is hosted on Google Kubernetes Engine with 4 pods with each pod having 2 high performance CPUs on the cloud.

5 Components

This section will provide an overview of the components:

5.1 Data Pre-Processing

As the original data obtained from Kaggle had a lot of unrequired rows, we had to clean the data and convert it to a form that was usable for us. To achieve this, we had to import the original data into MySQL and execute a couple of queries so as to as generate a new csv file which contained data such as **purchase date**, **appliance**, **capture_date**, **energy**, **warranty_date** as these are the columns that are required for us to predict the age for a given appliance.

Once the necessary file was generated, we use this file to create our spark dataFrame structure wherein we use this file as the input and create a dataFrame with columns **purchase_date**, **appliance**, **capture_date**, **energy**, using this we create our testing and training data set, and attach a **uniqueid** to each appliance so that it can be easily identified. And finally this new structure with fileName flag **training** is then pushed into as training data into **Cassandra** which will then be used by the algorithms to predict the appliance age.

5.2 Kafka

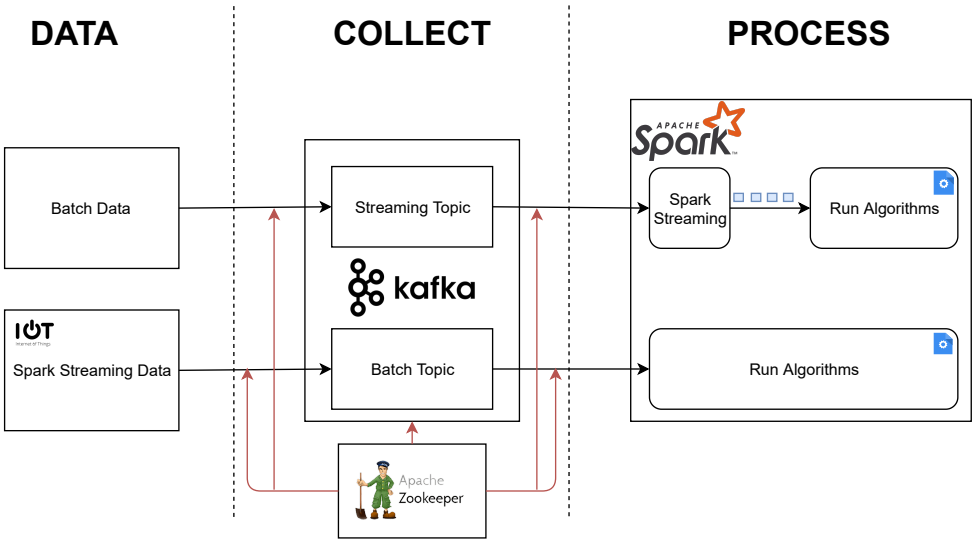

Figure 3: Kafka

Appliances emit data continuously, and as the presumed source of data is from multiple IoT devices for our application. It makes the processing of streaming data very critical. Spark offers a solution to handle streaming data with the use of Spark Streaming API, which enables scalable, high-throughput, fault-tolerant stream processing of live data streams. However, how do we transfer these data in a scalable and fault-tolerant way to Spark? This is where Apache Kafka comes into play. Kafka is a potential messaging and integration platform for Spark streaming. Kafka acts as the central hub for real-time streams of data and is processed using complex algorithms in Spark Streaming.

Along with Kafka, we also included Zookeeper as it is a must by design. Zookeeper has the responsibility of managing Kafka cluster and tracks the status of Kafka cluster nodes. It also keeps track of Kafka topics, partitions, and brokers.

5.3 Database - Cassandra

As the application processes IoT related data, Cassandra database was selected. It helps in seamlessly handling streaming data and can be easily scalable. Figure 4 represent the tables that are used. *training* table contains data that is used as the training set for the algorithms. Next, the *prediction_ knn* and *prediction_ kmeans* tables are used for storing the KNN and KMeans output. Finally, the *mst_ appliance* table is used for reporting as the data stored in all other tables have the appliance in the discretized form.

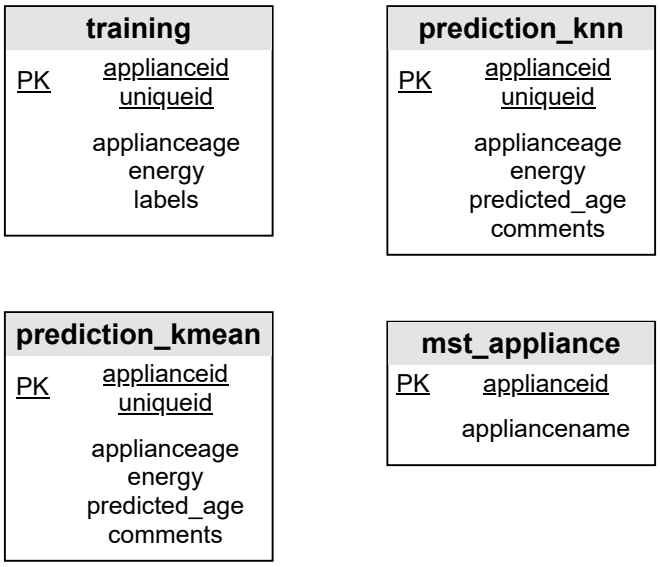

Figure 4: Cassandra Tables

5.4 Spark - Application workflow

Figure 5 represents the workflow for the application. The application handles two types of prediction algorithms: KNN (K Nearest Neighbor) is a supervised algorithm, and KMeans, which is an unsupervised algorithm. The primary motivation for implementing different algorithms is to compare their performance and to analyze the working of the algorithms

that are implemented using Spark SQL. For KNN, the training data requires no specialized training, and it is stored in the cache for faster processing. However, for KMeans, there exists a training phase of the data to generate a model.

Both algorithms can be executed using a stream or batching process. Few changes needed to be incorporated into the algorithms for these processes as for streaming data, Window functions cannot be used. Finally, after the algorithms are executed, the output will be appended to prediction_knn and prediction_kmeans table in Cassandra database for KNN and KMeans algorithms, respectively.

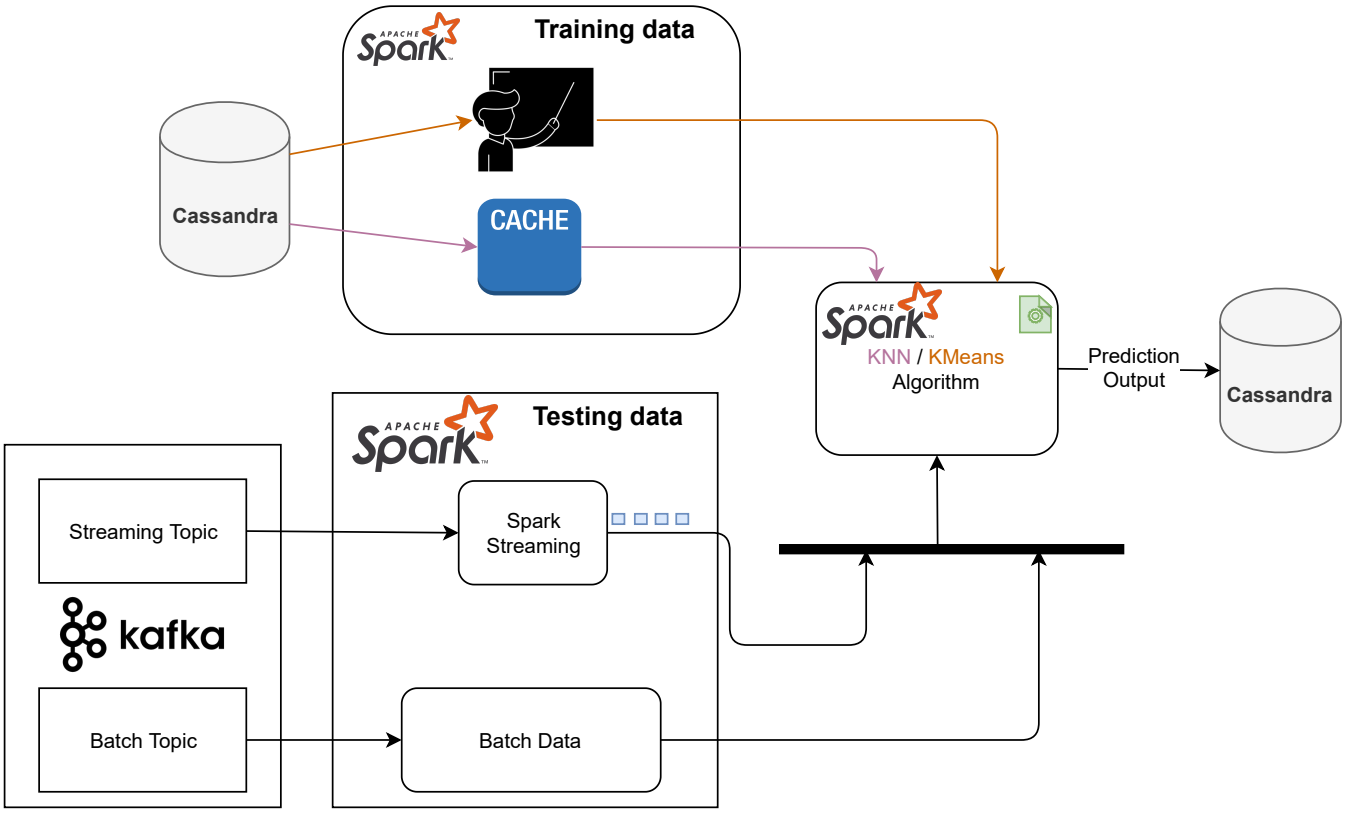

Figure 5: Application Workflow

5.5 Orchestrator

The underlying algorithm used by the application can be swapped externally, this is handled by a custom orchestrator class called Model. The structure of the Model class is as shown below,

```
case class Model(algorithm: String,
                 trainingDataFrame: DataFrame,
                 k: Int = 1,
                 continuousTraining: Boolean = false,
                 isStreaming: Boolean = false,
                 maxItr: Int = 1)
```

The variable **algorithm** is currently used to swap the underlying algorithm to either KNN or KMeans that will be used to predict the appliance life time. The choice of algorithm is externally handled through the use of "algorithm.type" key present in application.config file, "knn" or "kmeans" are the two allowed values that can be used as the valid values to this key.

The Model also expects the training dataframe to be supplied, this is provide the developers with enough flexibility to pre-process their training data if required.

"continuousTraining" is currently an experimental flag that will determine potential interesting training data from the novel data. After every prediction, the result is analysed to collect records that can widen the perspective of the existing training data, the records that qualify some conditions will be added to the training data and will be used in the subsequent prediction phases. The functionality can only be validated for correctness based on its performance over a period of time.

isStreaming is a flag set in the orchestrator to indicate the kafka channel through which the novel data will arrive for prediction. If set to "true", the **stream query channel** will be consumed, otherwise the **batch query channel** is consumed.

The orchestrator invokes train() and predict() functions belonging to the respective algorithm to control the flow of the application as and when necessary. The underlying framework is decoupled from the algorithm, this enables us to introduce new algorithms without affecting the underlying framework.

5.5.1 KNN

K-nearest neighbors is one of the algorithms available to predict the appliance remaining life time. The algorithm strives to find the "k" near data points with respect to the novel data that requires prediction. The mean appliance age of the "k" near data points is taken as the reference age to predict the expected appliance life time of the novel data. The process is split into two distinct stages, training and prediction.

Training: In this phase the training data is provided by the ochestrator class (Model) which will be cached in the train method. Since KNN does not involve a specific logical training phase, we utilize this method to cache the training data in the memory to reuse it for multiple predictions.

Predictions: The code is purely built on SparkSql (DataFrames). The following steps are undertaken to perform predictions,

- The novel data is first joined with the training data based on appliance id. Unique id present in the novel data is used to trace back the predictions to the original input record.
- vectorTR and vectorTE are two columns that should be present in training and novel input data respectively. The Euclidean distance is determine between these 2 columns.

This design decision is taken to ensure that the any number of feature columns can be added or subtracted from the algorithm without affecting the underlying framework and it is easy to understand.

- The resulting DataFrame is partioned by "uniqueId" and is sorted in descending order based on the distance computed in step 2, row number is added to tag the rows based on the descending order.
- Row numbers greater than the "K" value is filtered out from the DataFrame as they are not the "k" near neighbours.
- Group the records based on their "uniqueId" and determine their mean appliance age, utilize this information to perform the prediction.

$$predicted\_age = mean\_appliance\_age - novel\_appliance\_age$$

$$prediction = \begin{cases} \text{appliance has outlasted} & \text{if } predicted\_age <= 0 \\ \text{can survive for another predicted\_age} & \text{otherwise} \end{cases} \quad (1)$$

The records where predicted_age >= 0 will be added as new training data for continuous training functionality, as these are appliances fall outside the boundaries of our training dataset.

Performance: The performance of KNN is skewed towards the size of the training data. Lack of training phase in the KNN algorithm leads to easy transition towards a new training data, but is also a drawback of KNN as the training data will be refreshed on every prediction. Optimizations are implemented to avoid this by introducing caches to keep the training data in memory for quicker access. Formally we can say that the prediction time for KNN is constant and will take slightly more compute time in-comparison to KMeans or other prediction algorithms that involves a training phase.

5.5.2 K-Means

KMeans is another algorithm that can be used to predict the appliance life time in our application. The orchestrator parameter alogirthm has to be set to "kmeans" to enable this algorithm. The major difference of KMeans to KNN is the presence of an training phase.

Training: In this phase, the training data is split into "k" clusters and "k" centroid/prototype values are determined. The prototypes serve as reference points that will be used to predict the appliance life time of novel data. The following steps are undertaken during training,

- **Determine the K:** This step will only be performed once before the training begins. Rather than fixating on a single "k" value which is bound to change in our application, the count of distinct application id's present in the training data is used as the "k" value, as this is the ideal "k" value in our application. This is an optimization towards the accuracy of the application's predictions.

- "k" initial centroids/prototypes are picked at random for each appliance id and the data points closest to the controids are grouped. Euclidean distance is used to compute the distances. The algorithm is purely implemented on SparkSql.
- The mean of the "k" groups is determined to compute the new centroids/prototypes.
- The step 2 and 3 are repeated until any of the following stop criteria is satisfied.
  - Maximum allowed iterations are achieved.
  - There is no significant changes between the i^{th} prototypes with the $i-1$ prototypes.
- stop the training process and return the centorids/prototypes as a Dataframe.

Predictions: A cross join is performed between the novel input data and the prototypes computed in the training data. The distance between the novel data and the prototypes are computed and the prototype with the least distance to the novel data is chosen as the reference for prediction. The prediction is performed similar to the knn algorithm as shown in Equation 1.

The continuous training is also performed similar to the knn algorithm, where the records records that were currently out of scope of our training data is appended to the existing training data.

Performance: The KMeans algorithm spends majority of its time in the training phase, where it determines the ideal centroids for the given training data. Subsequent predictions performed by this algorithm can be done in record time as the centroids from the training phase is reused for predictions, unlike the KNN algorithm where the distances between the training and the test data were recomputed on each predictions.

5.6 Data Visualization - Tableau

Tableau software is a tool that helps make Big Data small, and the data insightful and actionable. In this current era, where data is considered gold, hence using a visualization tool to understand data is crucial.

As our project handles IoT house appliance energy consumption related data and predicting the lifetime of an appliance is interesting for data scientists to understand the trend in various appliances. With the help of Spark, our project predicted the ages of the six selected appliances, but without proper visualization, it is difficult to understand the data output. We had a look at some visualization tools like Grafana, but due to deployment constraints on the cloud had to drop it. Finally, the team was impressed with Tableau and its simplicity in creating and flexibility in designing a dashboard.

With the use of the dashboard, the users can view all the predicted outcomes from KNN and KMeans algorithm. Then they can view graphs that provide information on the count of appliances that have outlasted the training set and the number of months expected for the rest. Figure 6 represents the main dashboard page with filters to select the appliances

that the user wants to analyze. The two tables KNN and KMeans is the complete outcome of the prediction algorithm. The bar charts on the right-upper side provide visual output.

The second tab shown in Figure 7 provides a view of the average age of all the six appliances that have been predicted using both KNN and KMeans algorithm. This will be interesting for researchers to understand the accuracy of both algorithms.

The data refresh rate for these dashboards can be adjusted, and the users can choose if they want to view extracted data or live data from Cassandra based on the refresh frequency period. To make this dashboard accessible to a broader audience, the team has also released it on Tableau Online.

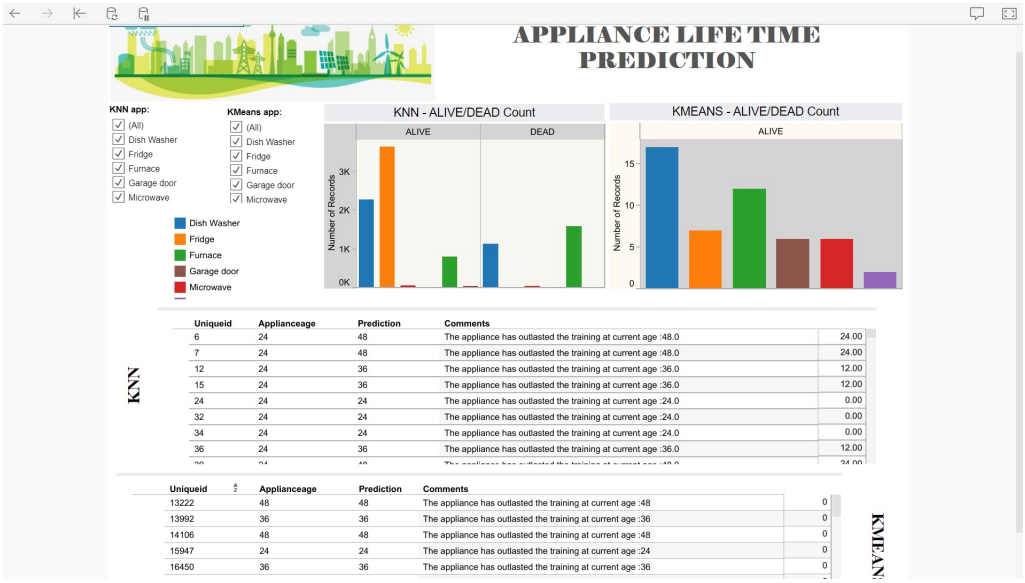

Figure 6: Tableau - Main Dashboard

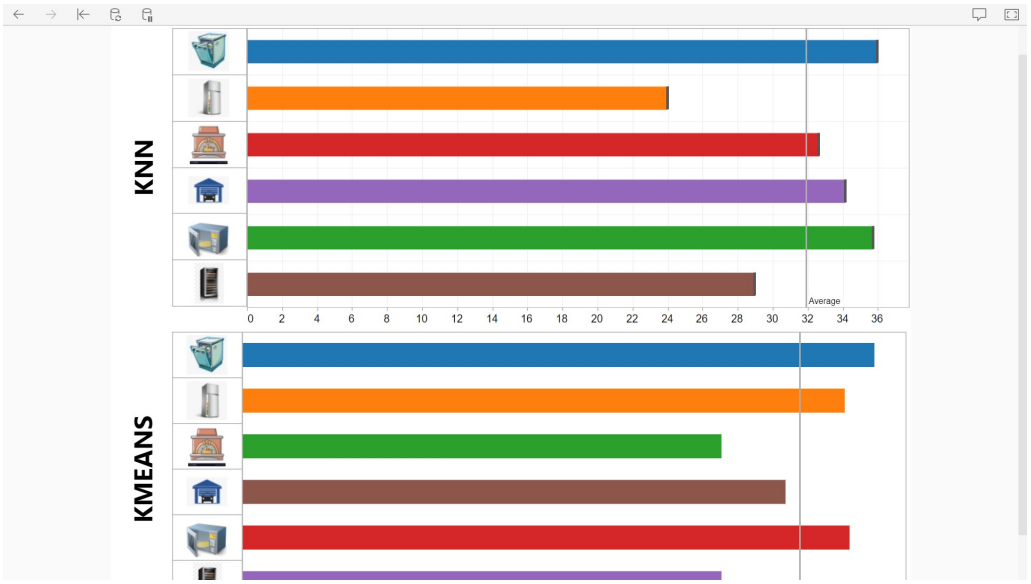

Figure 7: Tableau - Average appliance age