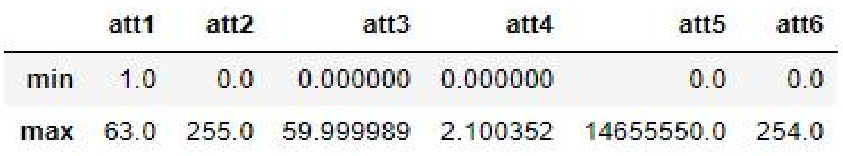
Fraud and Anomaly Detection: apply the unsupervised anomaly detection techniques to implement and deploy a model for real-time anomaly detection.

In this report, anomaly detection was implemented using k means algorithm with the help of sklearn. Starting with simple data preprocessing then experimenting to finding the right parameters for the K means algorithm and comparing the performance of the model trained on subsection of the features. Note that all experiments are done with the default values that sklearn provide unless indicated otherwise.

Data preprocessing:

We scale the data when the values of the features are so far from each other, and it's crucial especially when using that compute distance between data like K-means. for example if we have data describing house prices, the first feature we have is the area of the house, for example, their values range between 300 – 900 square meters, and the second feature is the number of rooms and their values range between 1-10 rooms and the third feature is the price in dollars, so the values range Between 100,000 and 500,000, we notice the difference in the measurements and thus the difficulty of working the model, so that is why we use scaling to make the values close to each other, ranging between zero and one.

What we are interested in here is the effect of difference and dimension of these values on the k-mean algorithm because it uses the distance between data points to determine the similarity between them. When looking at the data that we have, we find that:



	att1	att2	att3	att4	att5	att6
min	1.0	0.0	0.000000	0.000000	0.0	0.0
max	63.0	255.0	59.999989	2.100352	14655550.0	254.0

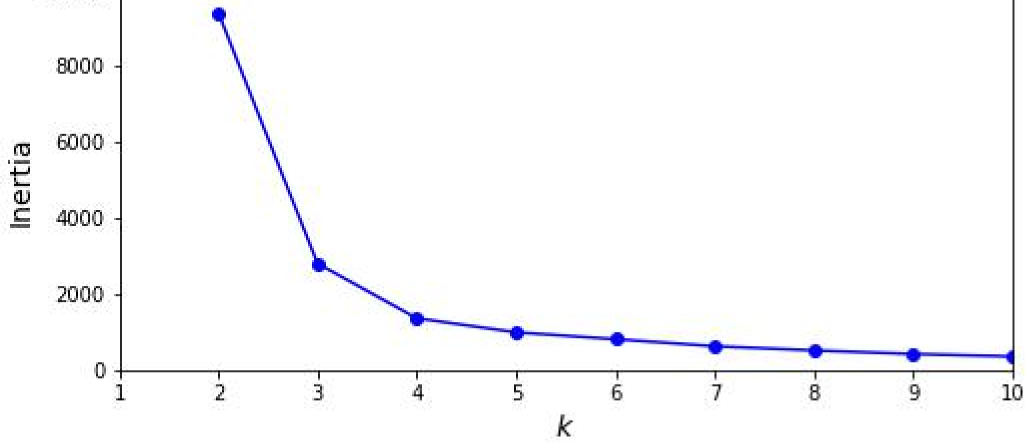
Notice that the range of att1 between (1 - 63) and the range of att2 (0 - 255) and to the last feature the range is varying and far from each other. we will use normalization, which is subtracting the value from the smallest value divided by the difference between the highest value and the lowest value, as the resulting value is in the range between 0 - 1, and by this we will bring the values closer together

$$X' = X - Xmin / Xmax - Xmin$$

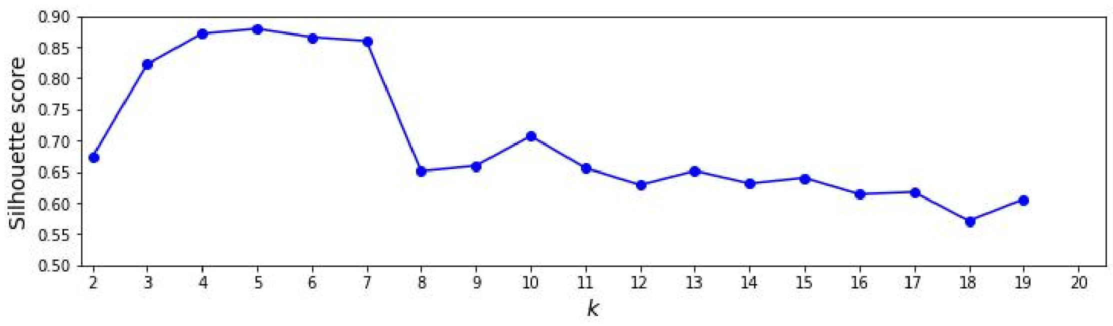
We need to divide the data into training and test data in order to increase accuracy and to avoid mislead the results when using the same data in training and test, the model gives high accuracy and that is a wrong result of course, because the model performs a process similar to the preservation process and will not give correct results in new data.

Finding the best parameters for the model:

Finding the best number of clusters is no easy task. For instance picking the model with the lowest Inertia value (internal evaluation method) does not guarantee the best K value because inertia can get very low when you have many clusters (more than necessary). (Geron 2017) so some useful technique to find the optimal k is to use the elbow method. (Geeksforgeeks n.d) The basic idea behind this method is that you plot the inertia as a function of k (see figure). The point where this distortion declines the most is the elbow point. And that point can show you the value that reduces k inertia the most in small values of k.



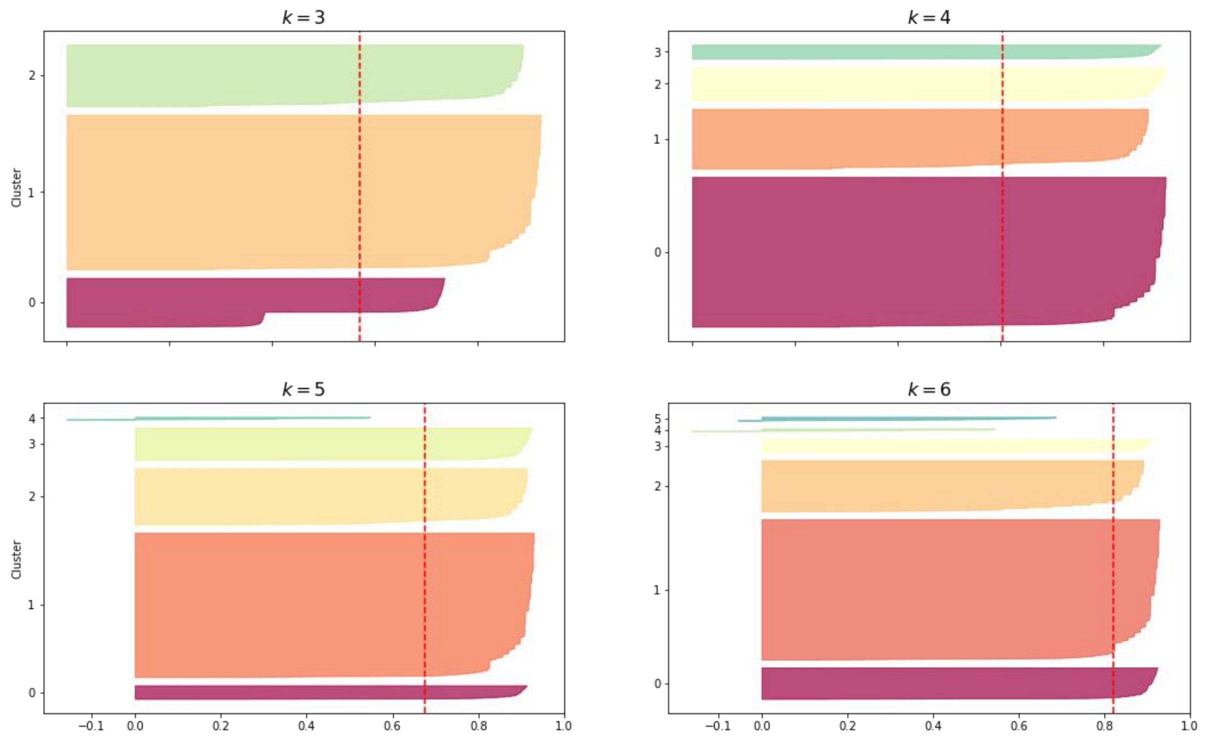
The elbow point is clearly **K = 3** this method is useful and easy but to ensure that this is the right value of k even further we could use a better method (but more computationally expensive) which is silhouette score which measures the distance between each data point and the centroid of the cluster it was assigned to and the closest centroid belonging to another cluster. This value ranges between -1 and +1 for each instance and is called silhouette coefficient. The higher the value the better.



By investigating the plotted values of silhouette score for 19 values of K using the built in sklearn *silhouette_score* method we can see that k 4,5,6 and 3 got the highest of scores and they are close to one another

```
3 --     0.8227183088905957
4 --     0.8717928384890301
5 --     0.8797320267828675
6 --     0.8660770731011901
```

So we can carry our analysis further By using an even more informative visualization called silhouette diagram for the k = 3,4,5,6 to choses the best value between the good candidates. You can obtain this diagram by plotting every instance's silhouette coefficient, sorted by the cluster they are assigned to and by the value of the coefficient. (Geron 2017)



The vertical read lines represent the silhouette score for each number of clusters. When most of the instances in a cluster have a lower coefficient than this score (i.e., if many of the instances stop short of the dashed line, ending to the left of it), then the cluster is rather bad since this means its instances are much too close to other clusters. We can see the cluster number 4 in k = 4 and cluster 4 and 5 in k =5 have most of its instances lower than the silhouette score which indicates it was an unnecessary cluster. And comparing k=3 and k= 4 looking at cluster 0 in k=3 have some of it's instans lower than the score. and it appears **K =4** doesn't suffer the same problem and should be a better choice of K.

After identifying the best k from the previous step. It's an easy task to compare models with different parameters. Will use external evaluation metrics called homogeneity scores which measure whether each cluster contains only members of a single class. This metric implemented using sklearn homogenetiry_socre function. In order to apply it will use the true labels of the data. (Scikit-learn n.d)

The baseline homogeneity score for the default values of KMean is 0.568870. Although KMean implementation in sklearn doesn't have a bunch of parameters to play with. It has some parameters that will define the way the model will be computed. for examples init parameters which control the

Method for initialization can be either set to ***‘k-means++’, ‘random’.*** *setting it to random* reduces the performance to 0.5494.

Finally experiment with changing parameters like “tol” which represent tolerance with regards to Frobenius norm of the difference in the cluster centers of two consecutive iterations to declare convergence. But most of this parameters didn’t affect the value of homogeneity of the model.

Feature analysis:

The number of features is relatively small. But we can compare different pairs of features to get an even faster model using Recursive Feature Elimination (RFE). We got feather “att2” and “att5” as the best pair of features to use. But in term of accuracy they scored 0.46729 which is worse. But sure this drop in accuracy were gained in speed. As we know the time complexity of KMean is

$$O(nkId)$$

Where n = number of points, k = number of clusters, I = number of iterations, d = number of attributes.

So we did a time profiling using (%%timeit magic function) to fit 2 models one with full features and the second with only the two features.

For the full model: 512 ms per loop
For the 2 features model: 380 ms per loop

Therefore, it almost halving the time recurred for the model to fit the data and still maintaining a good cluster so it might be used in an online system if we want a very quick response time.

Streaming k-means vs k-means and bisecting k-means for streaming datasets with respect to Spark:

K-Means Algorithm has a few limitations, t only identifies spherical shaped clusters, it cannot identify, if the clusters are non-spherical or of various size and density. In addition, it suffers from local minima and has a problem when the data contains outliers.

Bisecting K-Means Algorithm is a modification of the K-Means algorithm. It can produce partitional/hierarchical clustering. It can recognize clusters of any shape and size. And It beats K-Means in entropy measurement (Entropy measurement is a measure of the randomness in the data being processed). If the entropy of the given data, being processed is high, it is difficult is to draw conclusions from that data.

Examples to show the different:

If you walk into a hall consisting of many chairs, and sit on, one of them without observing any other chair in particular or thinking if you could hear to the speaker in the stage from the chair you are seated. This kind of approach can be called '**K-Means'**.

If you walk into a hall consisting of many chairs. You observe all the chairs, and decide which one to occupy; based on various factors like-if you could hear the speaker from that position and if the chair is placed near the air conditioner. This kind of approach can be called '**Bisecting K-Means'**.

Spark implement k-mean and implementation includes a parallelized variant of the k-means++ method called kmeans||. The implementation in spark.mllib has the many parameters *like* number of desired clusters. Maximum number of iterations to run. *Initialization Mode* specifies either random initialization or initialization via k-means||. The number of steps in the k-means|| algorithm, distance threshold within which we consider k-means to have converged and *initial Model* is an optional set of cluster centers used for initialization. If this parameter is supplied, only one run is performed.

Bisecting K-means can often be much faster than regular K-means and have this parameter when implanting it in spark.mlib, desired number of leaf clusters, the max number of k-means iterations to split clusters.

When data arrive in a stream, we may want to estimate clusters dynamically, updating them as new data arrive. spark.mllib provides support for **streaming k-means** clustering, with parameters to control the decay (or "forgetfulness") of the estimates. The algorithm uses a generalization of the mini-batch k-means update rule. For each batch of data, we assign all points to their nearest cluster, compute new cluster centers, then update each cluster.(geeks fo geeks n.d)

Conclusions:

I rescale the data from the original value to new values between 0 and 1 to make it easy to handle using subtracting the value from the smallest value divided by the difference between the highest value and the lowest value and then split the data after that I have used elbow method to find the optimal k value and the concept is plot the inertia as a function of k. The point where this distortion declines the most is the elbow point. And that point can show you the value that reduces k inertia the most in small values of k.To be more precisely I used more informative visualization called silhouette diagram to choses the best value between the good candidates I have got from last method and I find that the k=4 it's the best choice .Finally the number of feature is already small but I had compare some pairs of feature trying to get faster model using Recursive Feature Elimination i got feature "att2" and "att5" as the best pair of features to use. But in term of accuracy they scored 0.46729 which is worse. Sure, this drop in accuracy were gained in speed.

Reference:

Scikit-learn S n.d, Machine-learning tutorial, viewed 1 May 2021, <https://scikit-learn.org/stable/modules/clustering.html#homogeneity-completeness>

Geeks for geeks S n.d, clustering , viewed 1 May 2021, <https://www.geeksforgeeks.org/ml-determine-the-optimal-value-of-k-in-k-means-clustering>

Geron, A. (2017). *Hands-on machine learning with scikit-learn and TensorFlow*. Sebastopol, CA: O'Reilly Media.

Geeks for geeks S n.d, Bisecting K-Means Algorithm Introduction
, viewed 1 May 2021,
< https://www.geeksforgeeks.org/bisecting-k-means-algorithm-introduction/>