

©

Amanpreet Singh
All rights reserved

If you can't explain it simply, you don't understand it well enough - Einstein, the
Man and His Achievement By G. J. Whitrow, Dover Press 1973.

TABLE OF CONTENTS

Acknowledgements	iii
Table of Contents	iv
Chapter I: Introduction	1
Chapter II: Problem 1-Anomaly Detection	2
Chapter III: Feature Engineering for Computer Vision	
Problem 2-Features for SIGN Language Detection	15
Chapter IV: Statistics in Finance-Problem 3 Stock Forecasting	30
Chapter V: Statistics and Artificial Intelligence Optimization	50

Chapter 1

INTRODUCTION

We have an arsenal of Statistics techniques that are very helpful in Data Science if are used appropriately. The aim of this research is to experiment with Real world problems and have in depth review of different Statistics techniques to figure out which ,what and how a certain technique could be useful in different areas of Data Science and how can be used in practical real-world problems. I will do a depth review of when and on what type of situation and dataset, a certain Statistical method could be implemented in real-world scenarios. I will also be researching multiple decisions taken for selecting an appropriate Optimizer and Activation functions for Deep Learning Models, based on the type of data that we can only understand after the Statistics Analysis of the data.

I have seen that most people have Statistics knowledge but do not know the actual practical implementation of the same, which makes the knowledge useless. So in this research, I will explore and implement exact Statistics methods to solve specific real-world problems efficiently.

In this research, I will take some Real world problems and I will try to solve them, by researching and answering various questions on Selecting a Statistical method to appropriately implementing them, I will look in depth for various differences in different techniques used for similar purposes and research, why and when a certain technique is more useful in certain cases and what are the major differences.

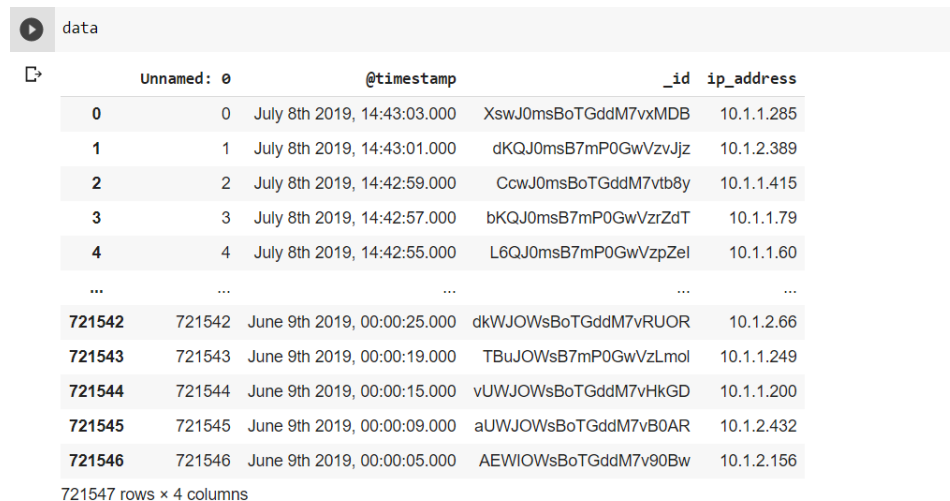
All codes and Datasets can be found on this Github repository:

<https://github.com/Alexamannn/Critical-Review-of-Statistics-in-Data-Science-and-Machine-Learning>

Chapter 2

PROBLEM 1-ANOMALY DETECTION

Anomaly detection, also called outlier detection, is a step in data mining that finds points in data and observations that are different from a dataset's normal behavior. Anomalous data can be an indication of things in data like technical glitches, malfunctioning of a module, potential opportunity, change in customer behavior, etc. According to statistics, we can say that an anomaly is an observation that is so far away from other observations that we can suspect it was generated by another means. Problem: IP-profiling, In this real-world problem, this data set is a logs data from a remote server generated over 15 days, showing the requests sent from various IPs to remote servers



	Unnamed: 0	@timestamp	_id	ip_address
0	0	July 8th 2019, 14:43:03.000	XswJ0msBoTGddM7vxMDB	10.1.1.285
1	1	July 8th 2019, 14:43:01.000	dKQJ0msB7mP0GwVzvJjz	10.1.2.389
2	2	July 8th 2019, 14:42:59.000	CcwJ0msBoTGddM7vtb8y	10.1.1.415
3	3	July 8th 2019, 14:42:57.000	bKQJ0msB7mP0GwVzrZdT	10.1.1.79
4	4	July 8th 2019, 14:42:55.000	L6QJ0msB7mP0GwVzpZel	10.1.1.60
...
721542	721542	June 9th 2019, 00:00:25.000	dkWJOWsBoTGddM7vRUOR	10.1.2.66
721543	721543	June 9th 2019, 00:00:19.000	TBuJOWsB7mP0GwVzLmol	10.1.1.249
721544	721544	June 9th 2019, 00:00:15.000	vUWJOWsBoTGddM7vHkGD	10.1.1.200
721545	721545	June 9th 2019, 00:00:09.000	aUWJOWsBoTGddM7vB0AR	10.1.2.432
721546	721546	June 9th 2019, 00:00:05.000	AEWJOWsBoTGddM7v90Bw	10.1.2.156

721547 rows x 4 columns

Figure 2.1: Logs Database

IP profiling: The Internet is huge and complex in structure which needs efforts for security and management. For network security and management ,it's necessary to have a better understanding of the traffic caused by different elements (IP hosts). The task to handle this problem is to study the behavior of IPs and find the anomalies or the IPs that have different patterns from the normal data set. To handle this problem, it is required to use the appropriate Statistics techniques that can help us in solving it.

Feature Engineering-Statistics is the heart of the most important step in Data science,

ie Feature Engineering. In order to create multiple variables that could help us analyze the patterns among different IPs, Statistics is required to find the distribution of variables along with the dataset, to understand if variables are following Normal distribution or are skewed. We also need to check the correlation among different variables created, so we can remove multicollinearity from the data. If there is multicollinearity, then variables that do not have correlation will be less precise, and there will be a bias in the data. Also to create new variables, knowledge of taking the mean of the variable and comparing to the maximum of the same, can help in outlier detection.

Normalization of data(Standardization)-Standardization is one of the Statistical methods for Normalising the data. In this, we put different variables on the same scale which allows comparing scores between different variables like weight, money, average hours, etc. To standardize variables, Standardization outputs scores that tell the number of standard deviations above or below the mean under which a specific observation of a variable falls. For example, a standardized value of 4 shows that observation is at 4 standard deviations above the mean. This interpretation will be true instead of the type of variables that we standardize. I created various news features, some of them are ratios, some of them are averages and some of them are Sums, In order to bring all these variables at the same scale, I performed Standardization.

Why is Normalization important?

Any method depending on distances to measure similarities between observations requires Normalization. If not Normalized then features with a high range will have a major influence. Hence, Normalization is important. For example Clustering, LSH, Support Vector Machines, etc.

For Measuring Variable Importance Variable importance tells that, how much a given model put more weights one variable to make accurate predictions. If the model uses a variable more to make a prediction then that variable is important for the model. In most models, the importance of features is calculated by comparing the absolute value of their coefficients. But, if the features are not normalized, comparing their coefficients becomes meaningless.

Penalty and Overfitting LASSO and Ridge regressions put a penalty on the coef-

ficients associated with each feature and the scale of variables affects how much penalty is applied to the coefficients. Because feature coefficient variables with more variance are small and will get less penalized. So normalization is required before fitting regressions.

	ip_address	total_count	daily_counts	is_weekend_ratio	td_mean	td_max
0	10.1.1.1	1446	40.0	2.070064	28.999308	362.0
1	10.1.1.100	2860	78.0	2.177778	14.427072	185.0
2	10.1.1.101	1465	40.0	2.191721	28.520492	211.0
3	10.1.1.106	1408	35.5	2.229358	29.771144	319.0
4	10.1.1.109	1459	42.5	2.206593	28.711934	278.0
...
381	10.1.2.86	4307	111.0	2.209389	9.441013	122.0
382	10.1.2.89	2826	73.5	2.196833	14.612743	207.0
383	10.1.2.90	2904	76.0	2.334099	14.215639	159.0
384	10.1.2.95	2868	77.5	2.204469	14.407394	188.0
385	10.1.2.99	1423	37.0	2.197753	29.407173	267.0

386 rows × 6 columns

Figure 2.2: Feature Engineering

New Features/Variables created After Feature Engineering

Total_count: Total Number of times an IP sent a Request in whole dataset

Daily_counts: Number of Daily requests by each IP

Is_weekend_ratio: Ratio of requests among the requests sent during the week and the requests sent during weekend

Td_mean: Average time a particular IP take between two requests

Td_max: The max time is taken by a particular IP to send a request between two requests.

Collinearity-Correlation: In order to find outliers, I will group similar IP addresses on the basis of new variables created, and Cluster will help me find out if there are some data points, which are Anomalous and outliers or not. But first, we have to check for Multicollinearity, as it can be a big problem in Clustering. Collinearity means that there is a high level of correlation between two variables and when more than two variables are involved it is known as multicollinearity. In cluster analysis, there are dependent variables or coefficients. Observations of variables are used for creating groups. Each observation belongs to one group, and each group can be

stated in terms of all the variables. When variables are collinear, some variables will get a higher weight than others. If two variables are perfectly correlated, it means they represent the same information, and that information will be represented twice in the data and will get twice the weight of all the other variables. so the final solution will be skewed, Thus it is important to check Collinearity among variables also in Clustering.

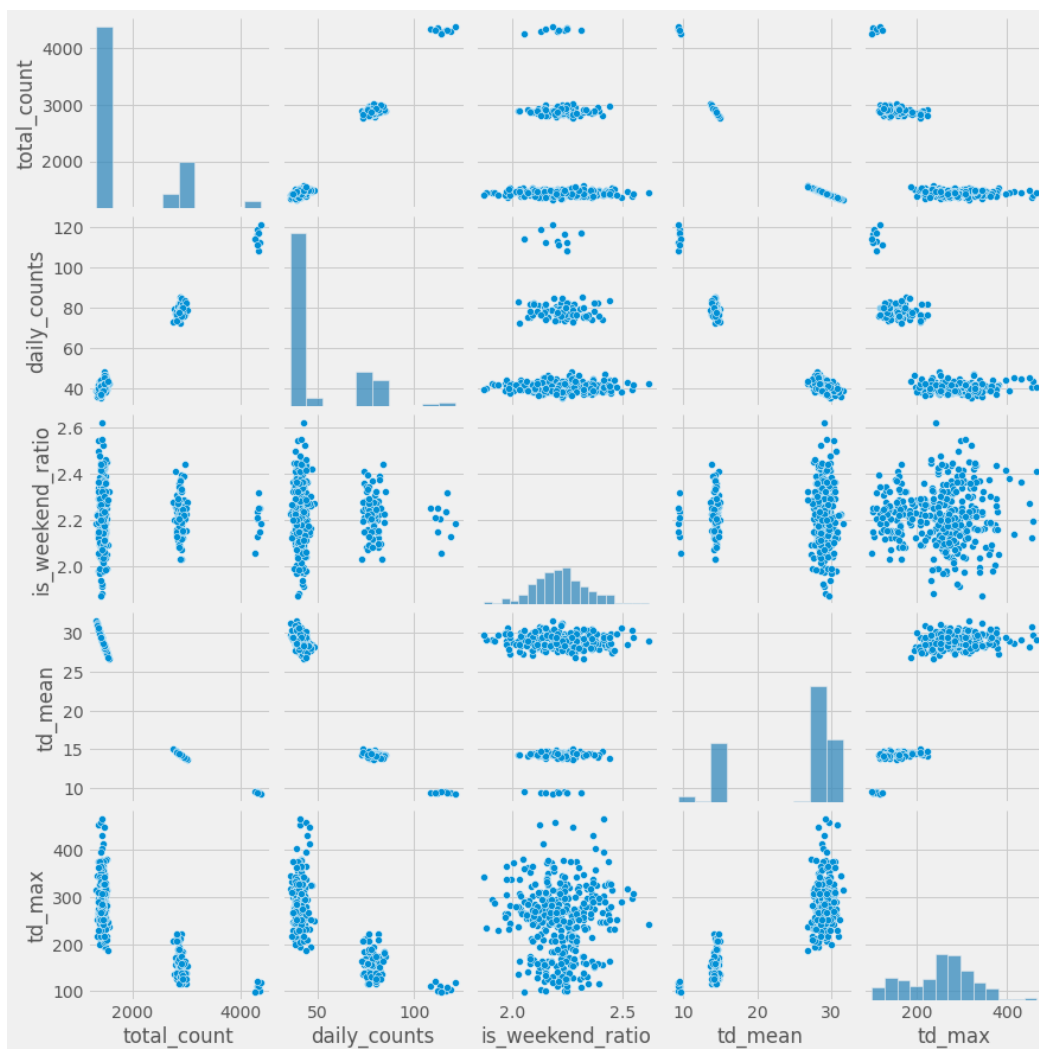


Figure 2.3: Correlation Matrix

After Feature engineering and Exploratory Data Analysis, I have my variables to find patterns in the data and detect outliers and anomalous data points. For this first I will try to group all the data points based on the features, There are many Clustering methods that work on the Statistics engine. I will use K means clustering for this problem. But question arises

Which Clustering Method is right? How to choose the Correct Clustering method?

Properties of data help to make the decision, so first, it is very important to be familiar with the data and the goals of the solution. if every attribute is equivalent or normalized), and linear, and the goal is to quantify data set and least square error is important in the analysis, then k-means can be a good option. K means is a better option one does not know the exact patterns in the raw data and wants to find insights. If one has a clear idea about the variables and their patterns and workings in a dataset, and have an idea of quantification of similarity and distance, and have a very good idea of how a variable will affect the whole analysis, then hierarchical clustering and DBSCAN are a good choice. If there is no way to know how to quantify similarity, then these options could not be good for the dataset. For example, RGB pixels in an image, here least square error could have importance and every attribute/variable is comparable with each other, so Kmeans is a very good choice on the other hand if it is a geographic data, distance is very useful and there will be patterns that one already may have the knowledge, so DBSCAN will be very good, on the other hand in the same problem if data is very clean then, Agglomerative Clustering is a better option

Different algorithms of Clustering are based on different statistical methods, For example Kmeans works on Centroids and Euclidean distance and can be of different shapes, DBSCAN works on Density and radius of the clusters and is useful when we know that there is a lot of noise and not important information in the data, and DBSCAN groups point with minimum support,i.e there should be the minimum number of points in one cluster, so if a point doesn't have those minimum number of partners to share the cluster, then they considered noise, whereas HAC works on bottom-up, where each point has its own cluster and then different clusters are grouped together to make one cluster,

One can also choose the Clustering method that shows the best results for the data

based on internal validity. Internal validity says that the more tight, dense are clusters from inside and have less density on outside then it has better internal validity. But one has to be careful as Clustering methods tend to over-fit because all methods try to maximize internal validity. On the other hand if one knows somehow the true clusters in the data, then the accuracy of clusters can be measured through external validity. One can also use different methods of clustering and then do Data Analysis by making clusters into categorical variables and see if the clusters are making sense or not and select the Clusters and cluster method that are making more sense according to the data, For example in business analysis, clustering different types of customers, if you using variables like profit, payment routine, and you get clusters between customers divided into two groups, with one gives more profit and pay timely and another group where customers profit are normal and also do no pay timely, then Clusters made make sense and the clustering method is valid.

For anomaly detection in this dataset, Kmeans was the best, as I do not have any idea of patterns in the data, and I am trying to create Labels for anomalies on the basis of Clusters. I want to find points on the basis of variance in the new features I selected and want the algorithm to group points, so I can find some different types of IPs and which IPs have similar usage can be grouped together in one cluster, and after I have similar IPs in the data, I can further make my strategy for finding anomalies or outliers in the dataset based on those groups usage patterns. In K means, we have to select the number of groups or in other words value for K. There are some methods to find the best number of K or groups for the dataset like the Silhouette Score, Elbow method. For this problem, I used the Elbow method, as it was defining good groups for the Dataset.

Elbow Method: Elbow method works on Sum of Squared Error and "Breaking Point". Squared Error for a point is the squared distance of a point from its Centroid. The method simply says that if the squared distance is low, means we are overfitting and eventually we will be giving each point its own cluster, and on the other hand if Squared Error is too high we are adding a lot of Bias in the clusters. So, the variance will go down with increased k. The elbow or the 'Breaking point' is where the curve will bend the most. We can think of Pareto Analysis in Economics and say to select the point where returns no longer add value to the extra cost. In clustering, we can say, one can choose a number of K so that adding additional clusters will not give much better modeling performance.

For our problem, I selected $k=5$ as in the image we can see, after 5 there is not so

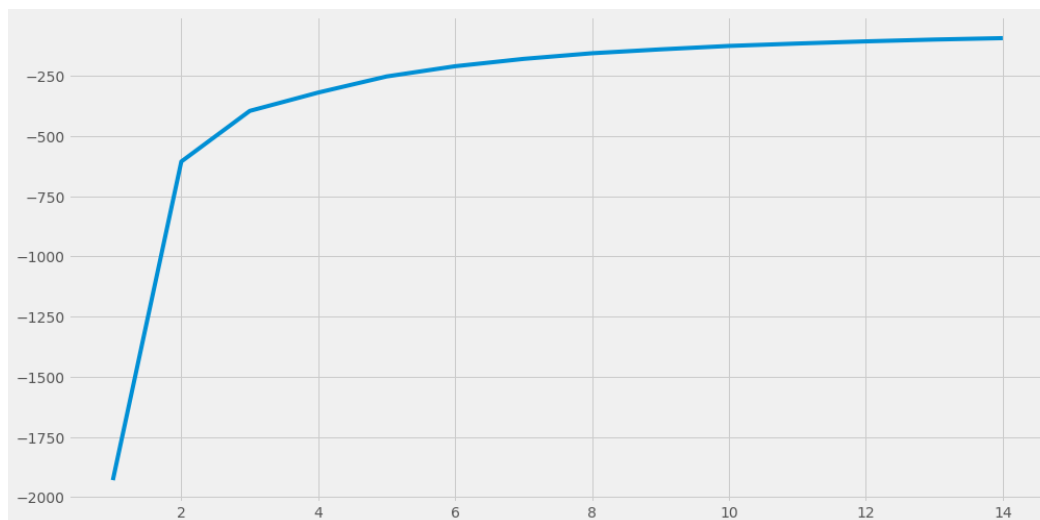


Figure 2.4: Elbow Method

much Squared error decline.

Now we have our clusters, Now in order to see patterns in the data, we need to somehow visualize the clusters, But data is high dimensional as it has 6 features, and to view clusters on a 2d graph, we need to reduce the dimensions of the data and make it 2d.

Here we have another Statistical technique to do it named T-SNE,

t-Distributed Stochastic Neighbour Embedding (Tsne): Tsne is a statistical dimension reduction technique that scales down the dimension almost similar to the original space, and makes it very good for visualizing data points separations. Tsne can be used for early visualization to understand the degree of separation of data points. Other dimension reduction techniques like PCA, SVD reduce data dimensions with variables projected on each other and dimensions in real disappear, which makes them not useful for data separation and good for visualisation.

The Task of TSNE is to take points from High dimensional space and search for a reliable representation of the same points in lower-dimensional space, The first step in the algorithm is to build a neighborhood graph between vectors of features based on distances

Then the graph connects feature vectors to their nearest nodes in terms of distance

in the feature space, and then builds a new representation of features on a 2d plane. For Model building, TSNE does a random walk on the feature graph. Tsne does not use Normal Gaussian distribution, rather uses t Distribution which wider and have longer tail than normal distribution, N in the image is a Hyper-

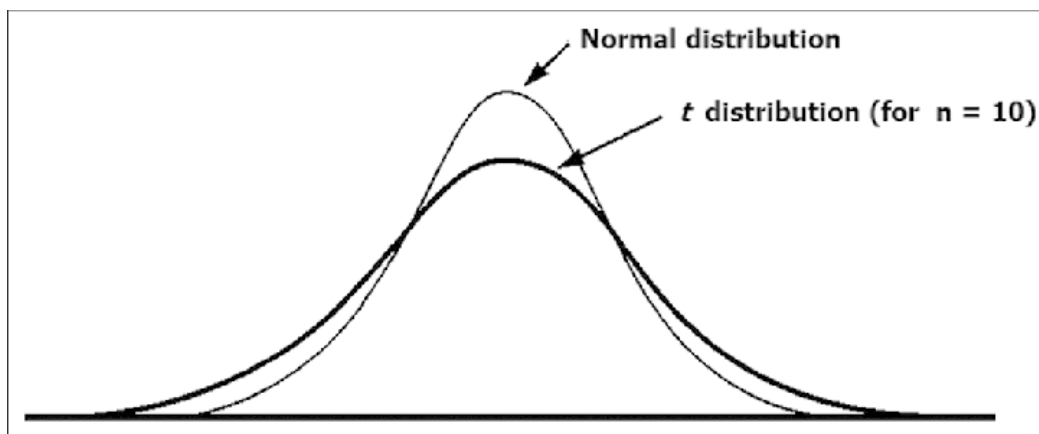


Figure 2.5: t Distribution Vs Normal Distribution

parameter in TSNE, named perplexity. Perplexity parameter is an estimation of the number of close neighbors each point has. The performance of TSNE is sensitive under different perplexities. In order to select the best perplexity, it mostly depends on the density of the data, so one can make an interpretation that a denser dataset may require larger perplexity. Mostly perplexity range between 5 and 50.

How does one determine the quality of the visualizations that TSNE made?

TSNE does not keep distances, instead, it retains probabilities, so measuring the Euclidean distances between the High dimensional point and the low dimensional point will be not meaningful. One method is to check Kullback-Leibler divergences which TSNE reports. It is acceptable to do t-SNE ten times, and select with the lowest KL divergence.

Tsne has the ability to create 2D graphs from data with hundreds of dimensions. But these graphs can be misinterpreted also. TSNE uses a nonlinear scale, which plots the points that are not similar to PCA. It could also be expected that TSNE can mix the dimensions as they are not distinct as in PCA data.

In the Anomaly problem ,for TSNE,I used perplexity =40 and gives KL divergence of 0.3241

```

▶ tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300, random_state=
tsne_results = tsne.fit_transform(scaledx)

[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 386 samples in 0.001s...
[t-SNE] Computed neighbors for 386 samples in 0.014s...
[t-SNE] Computed conditional probabilities for sample 386 / 386
[t-SNE] Mean sigma: 0.415830
[t-SNE] KL divergence after 250 iterations with early exaggeration: 54.996994
[t-SNE] KL divergence after 300 iterations: 0.324114

```

Figure 2.6: KL divergence TSNE

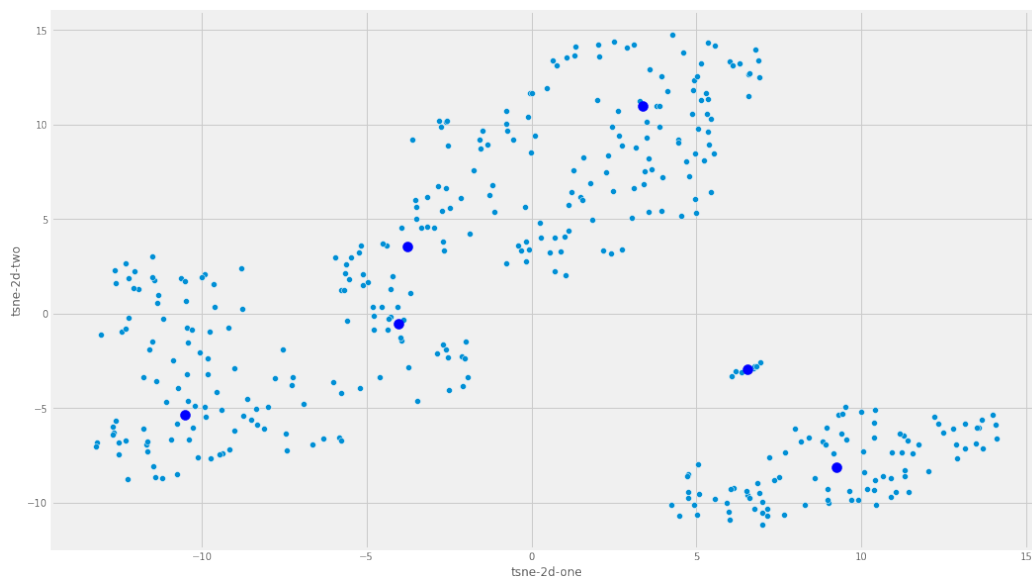


Figure 2.7: TSNE 2D Graph

Then I plotted K Means Cluster on the TSNE graph to see the patterns discovered by Kmeans clustering in the dataset

Now I have clusters and points on a 2d dimension, My strategy to find Anomalies in the data set is to find the sum of squared error between vectors of Centroids and all the features of an IP, The idea is to find the distance between two vectors, first is a vector of all the centroids and another is a vector of all the features of particular IP and then I will square it to add penalty according to Sum of squared error. The idea is to find out the data points or IPs, that have too different features from every centroids or cluster that define each group of IPs, and hence anomalies can be found. To find the distance between vectors, I will use another statistical technique named 2-norm of vector or in the case of matrices Frobenius Norm.

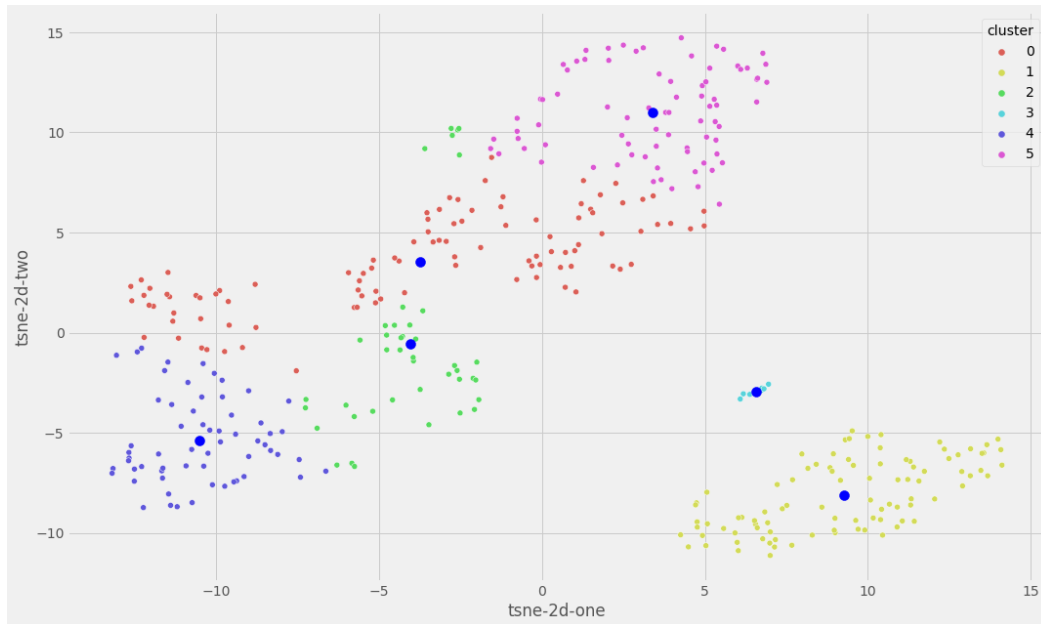


Figure 2.8: K-means Clusters on TSNE 2d graph

Vector and Matrix Norms: In order to find the magnitude or distance between matrices or vectors, or to decide which vector is bigger than the other, or in terms of matrices, which matrix can have bigger output or linear transformations, L2 or L1 norms are used. The norm of a vector is a way to find how much magnitude is of one vector from the other, or we can say how much distance is between these two vectors. The norm of a matrix is a way to find how large its elements are. It is a way of determining the size of the matrix irrespective of its number of rows or columns.

2 norm or Frobenius Norm The vector length is measured by the square root of the sum of the squares of the elements, also called Euclidean norm and in the case of matrices, it is defined as the square root of the sum of the absolute squares of its elements, 2 norm of a matrix is essentially, what that is is the square root of the sum over all elements of a matrix. It can be interpreted as what you get if you take a matrix and you reshape it into a really tall vector and then that will just be the 2 norm of that vector.

After finding the 2 norm distance between Centroids and All the features of an IP, I plotted the distance on a Histogram to find outliers or not normal patterns with the 2 norm, in other words, to find anomalies.

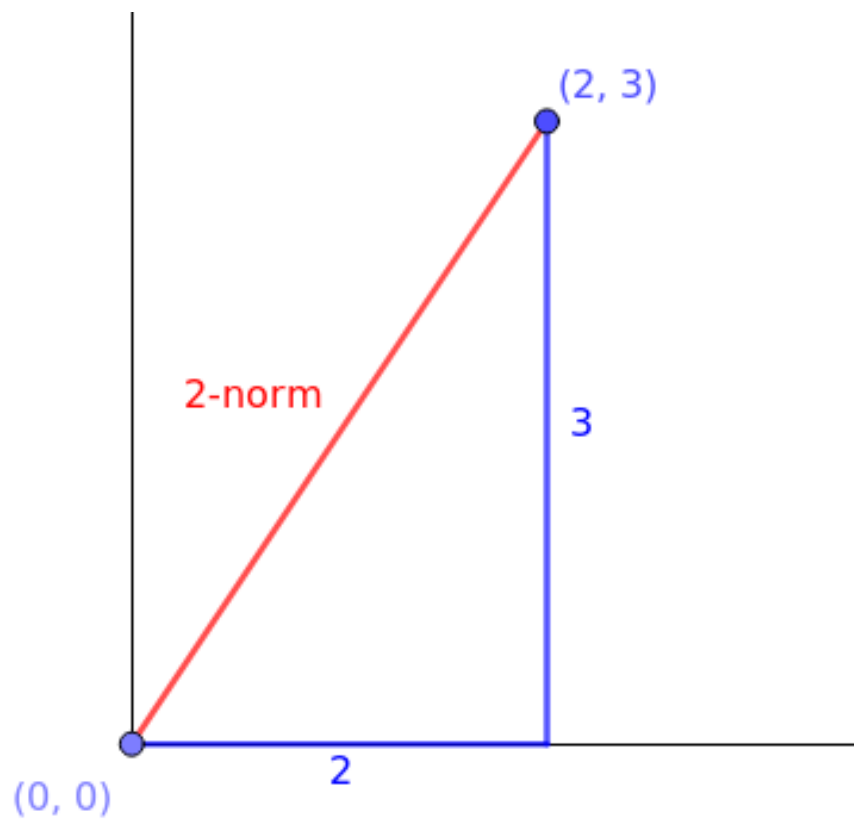


Figure 2.9: 2-Norm Distance

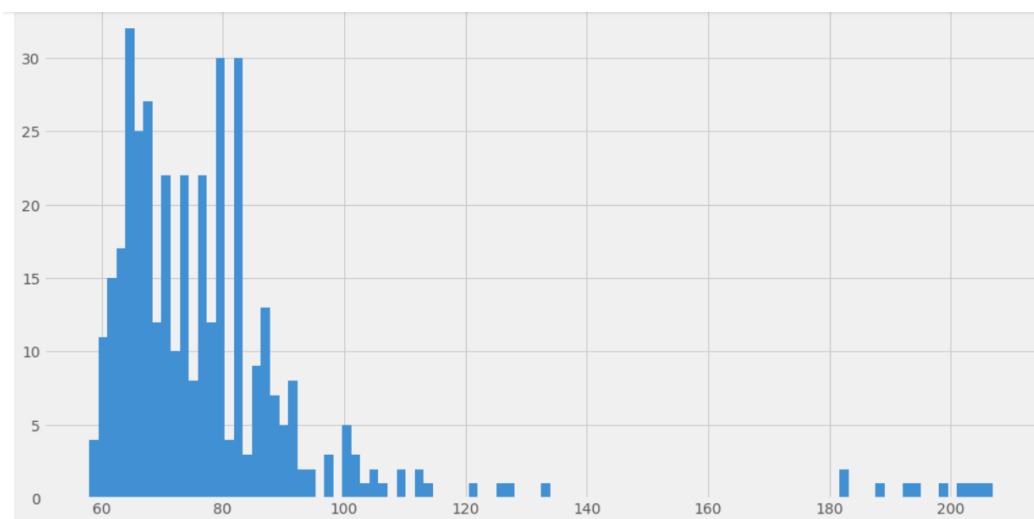


Figure 2.10: Distance on Histogram

As said before, according to statistics, we can say that an anomaly is an observation that is so different from other observations, that we can suspect it was generated by another means. We can see that after in the histogram, that all the normal patterns are before 110, and after 110 we can consider them outliers or anomalies, let's see how they look on the TSNE graph.

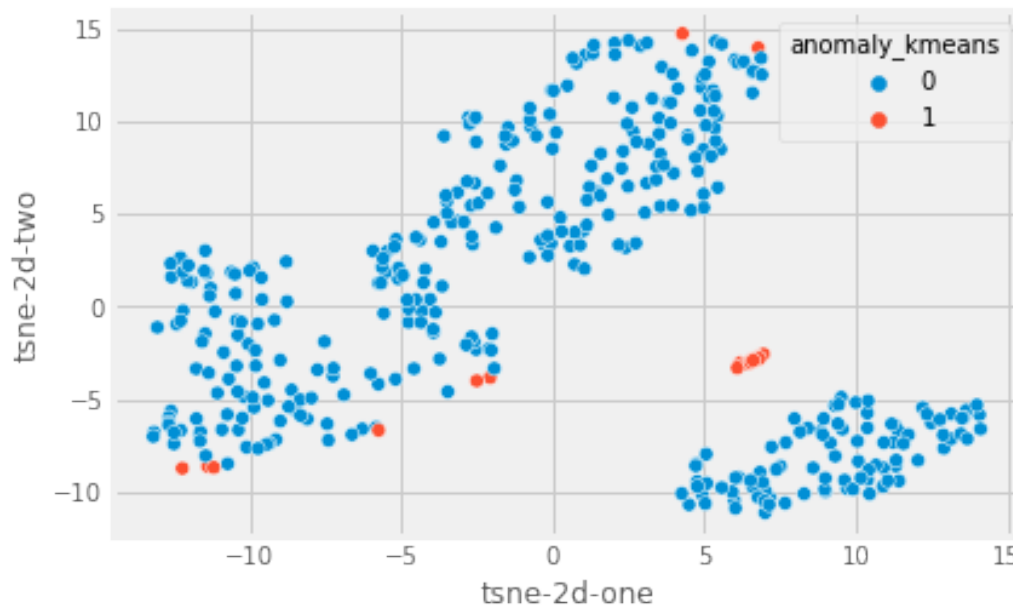


Figure 2.11: Anomalies/Outliers Detected

Hence we find the red points in the graph are the IP address that does not fit with the requests of the rest of the IP addresses

	total_count	daily_counts	is_weekend_ratio	td_mean	td_max	cluster	tsne-2d-one	tsne-2d-two	2norm	anomaly_kmeans	ip_address
14	1383	39.5	2.546154	30.306078	297.0	4	-11.403841	-8.626899	113	1	10.1.1.138
42	1365	40.5	2.123570	30.801320	455.0	2	-2.076416	-3.836896	121	1	10.1.1.199
62	4301	116.5	2.236268	9.459535	101.0	3	6.737013	-2.875531	199	1	10.1.1.249
70	1445	42.5	2.621554	29.042936	241.0	4	-12.239997	-8.738975	128	1	10.1.1.264
118	4300	118.5	2.127273	9.453361	104.0	3	6.177608	-3.054615	205	1	10.1.1.386
136	1438	40.0	1.881764	29.170494	235.0	5	6.776437	13.952807	110	1	10.1.1.427
164	4317	117.0	2.315668	9.417285	108.0	3	6.950425	-2.575877	202	1	10.1.1.486
177	4339	112.0	2.148766	9.368142	101.0	3	6.377236	-3.085320	195	1	10.1.1.163
188	4293	113.0	2.203731	9.456897	110.0	3	6.542520	-2.906541	189	1	10.1.1.186
253	1407	39.5	1.871429	29.747511	343.0	5	4.272238	14.723619	126	1	10.1.2.243
255	4353	112.0	2.250934	9.332721	102.0	3	6.746621	-2.773616	193	1	10.1.2.249
286	1408	43.5	2.192744	29.719261	458.0	2	-2.516750	-4.015271	114	1	10.1.2.323
311	4326	108.0	2.250188	9.392370	110.0	3	6.811514	-2.806853	183	1	10.1.2.386
331	1437	40.5	2.413302	29.176880	466.0	2	-5.757772	-6.683408	133	1	10.1.2.432
345	1428	41.5	2.552239	29.384022	308.0	4	-11.202197	-8.695792	113	1	10.1.2.463
357	4251	114.0	2.056075	9.571059	99.0	3	6.083502	-3.315783	203	1	10.1.2.486
370	4372	121.0	2.184268	9.268588	118.0	3	6.516491	-3.024493	207	1	10.1.2.63
381	4307	111.0	2.209389	9.441013	122.0	3	6.610925	-2.915438	183	1	10.1.2.86

Figure 2.12: Detected Anomalies

Chapter 3

FEATURE ENGINEERING FOR COMPUTER VISION PROBLEM 2-FEATURES FOR SIGN LANGUAGE DETECTION

Feature Engineering and Statistics

When we design our features, it is very important that the new features are a representation of original data, and in this case for Computer Vision, images, or videos. If we decide which type of information actually matters and can help us. For example in this problem of Sign language, we have to figure out hand movements in a sequence of images, to connect to a label, Label in this problem are words said in Sign Language. So for the perfect interpretation of any image or a video frame into tabular data, with exact information one require to solve the problem at hand in which Various Statistics methods and techniques should be used. Statistics help to get the real patterns in a data set, images, or Videos. For any kind of Learning algorithm, first, we require to recognize the data on pictures, then learn to understand it, and make predictions. Providing the data in a computer-readable format (as tables of numbers), one needs to transform the data in such a way to create meaningful features, so that algorithm has even less to figure out on its own. In various Computer Vision problems, in order to simplify the large data sets including images and videos, etc, with Statistics doing Feature extraction will help in reducing the resources and computational power and storage, required to describe a large set of data.

Problem at Hand

For Sign language Detection Feature Engineering,I took data from open-source video uploaded to help community on YouTube by scuolalatecnica for LIS ,and for I ll be collecting data for the word 'Cane' In this problem,I have to collect data in sequence of frames,as a word cane is as sequence of hands movements over different frames and I need to record the hand movements and convert them into meaningful interpretation using Statistical methods. In word Cane, I have sequence of hand movements over 43 frames i.e images.

Contour is a boundary around something that has well defined edges, which means that the machine is able to calculate difference in gradient (significant difference in magnitude of pixel value), In computer Vision Drawing a contour is based on Scan Line Polygon Fill Algorithm,This algorithm works by intersecting scanning lines

with edges of polygon and fill the polygon between pairs of intersections. Finding Hands position and tagging with contours,

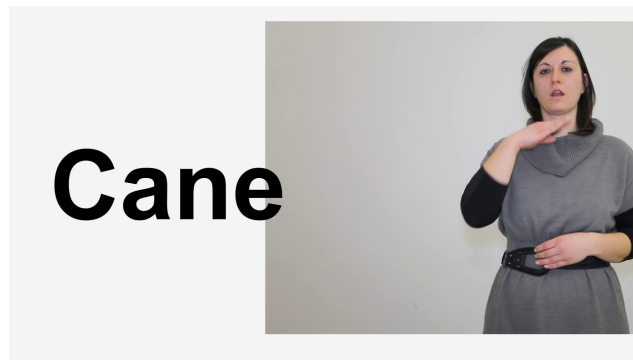


Figure 3.1: Sequence Frame Shot



Figure 3.2: Color Extracting

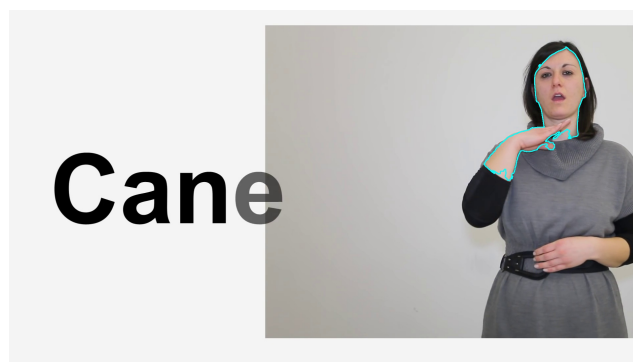


Figure 3.3: Contour Boundaries

The feature-based approaches required the detection and measurement of hand shape ,gesture and position using distances and angles,I have to find features or variables,that can define Hand position, shape, structure in complete sequence of the word Cane, such that those feature could be used for classification models and generating a Computer Vision Sign Detection app,

Convex Hull

In Computer Vision and Statistics, the boundary of a shape is called a “hull”. A boundary that does not have any concaves is “Convex Hull”. The convex hull of a set of points is found using OpenCV’s `convexHull` function.The convex hull, that is, the smallest multiple sides convex polygon that completely surrounds an object, In this case, convex hull in different sequences will wrap a convex polygon in different scenes from the start of the word Cane to the end of the word Cane, So we have recordings of hand visibility and shape-changing with the boundary that is Convex Hull



Figure 3.4: Hand Image

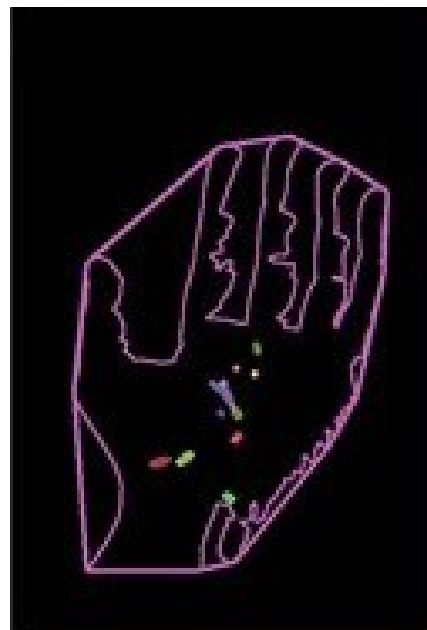


Figure 3.5: Convex Hull of the Hand

Now to make this feature useful, the problem is for each frame, I have multiple points in 2-dimensional space, plus I have 43 images. But to make them useful for actual analysis, I need to find a way to convert them into a low dimensional meaningful sequence of numbers, without losing useful information that helps map the Convex

Hull for a word Cane.

In statistics we have Multiple dimension Reduction Techniques, But the question arises

Which Dimension Reduction Technique to be used ? and Why? and In which cases different Dimension reduction Techniques cannot be used?

I am using various Dimension Reduction techniques in different features for this problem, and I will discuss why and for what reasons, this particular dimension reduction technique is mores useful in this case than others,

For Convex Hull, I used Principal Component Analysis(PCA) and Singular value decomposition (SVD) In convex Hull we have multiple 2D points for each images,so we have 43 sets of multiple 2d points that tell us about the boundary of the polygon



Figure 3.6: Dimension Reduction Process for Convex Hull points of 43 image sequence to SVD component

Why PCA for 2d Array of each Frame ,reduce to 1d array of each Frame?

Principal Component Analysis (PCA) We are all familiar with mathematics behind PCA, but the objective of the research is to find and rationalize the perfect usecase in the real world, for that I think, discussing PCA in non mathematical way is more appropriate to have an intuition of proper usage in real world.

PCA is suitable for ellipsoid in data. The ellipsoid is a wide variety of twisted circular shapes such as cigars, pancakes and eggs. All of this is neatly explained by the directions and lengths of their main (semi-) axes, such as the cigar or egg axis or the pancake plane. No matter how the ellipsoid is converted, the eigenvectors point to those key indicators and the eigenvalues give you the length. Very small eigenvalues are compliant with guidelines with little variability

Imagine opening a ice-cream store and we have 50 flavour of ice cream and want to figure out how to arrange them in the containers, so that the same taste buds are placed in the same container. There are a variety of ice cream flavors and textures - sweetness, bitterness, mint, fruity, strong etc. So the solution can be to put ice creams in categories to answer these two questions:

- 1) What are the most important qualities in identifying Ice cream groups?.for example does grouping based on sweetness make it easier to cluster ice cream into similar-tasting groups than grouping based on fruitiness
- 2) Can we narrow down our list of variables by including some of them? e.g. is there a variable that could be a mixture of "sweet and fruity and mint"

This is actually done by the PCA. The key elements are variables that define variability in the data set - in this case, which divides between groups. Each key component is one of the original descriptive variants, or a combination of some of your original descriptive variations.

In our use case, PCA will see the Boundary points of the polygon for Convex hull for each image. PCA will decide if it can cluster or group those points in a way that can be defined similarly in the spatial sense, for example What are the most important qualities in the points, in a spatial sense

Each dot in the "Convex Hull cloud" shows a specific location list, X and Y. Included. New items can be created by drawing a line in the center of the cloud and highlighting all the points in the line. This new property will be provided with a combination of lines The PCA will find the "best" line depending on the two different approaches

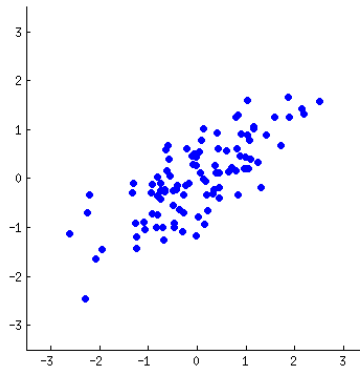


Figure 3.7: Exemplary Data Points

of what is "better". First, the price difference on this line should be large. Second, if we rebuild the first two elements from the new , the reconstruction error will be given by the length through Pythagoras theorem. In this way, the PCA stands for "key analysis" and the new asset is called the "primary component". And instead of "property" or "element" we usually mean "element" or "variable". Statistically, the variance/spread is measured as the average square distance from the center of the Convex Hull cloud to each new component. The sum of these numbers is equal to the distance between the center of the Convex Hull cloud and each blue dot; this is a Pythagoras theorem. Hence converting The X and Y axis at each point to another single point of choose Component by PCA, which has minimum error from the center and all other points. So, we will get a point in 1d array that will describe the X,Y points of the convex Hull point with most variance.

Now I have a matrix of 43 1d array of PCA component of convex hull that explains the most variance of points of the convex hull of the hand shape. In order to go further and reduce those 43 1d array to a sequence of 43 numbers that can be interpreted and translated together in a way that each point is a closest representation of the whole array of the frame according to all the arrays in population that are 43 different array. For this, we have a very useful Dimension Reduction technique named Singular Value Decomposition (SVD).

Singular Value Decomposition (SVD)

SVD can be thought of as compressing a matrix. It involves rotating a matrix and then compressing it horizontally and vertically, think of compressing a Unit circle,

into a egg shape ellipsoid. SVD can be defined as a way to combine data from multiple correlated vectors, and to create basis vectors that are guaranteed to be orthogonal in a high-dimension space, and define most of variance of the data. Singular Value Decomposition (SVD) provides a way to make the matrix into vectors and individual values. Similar to the way we make a whole number in its basic elements to learn by a whole number, we decompose any matrix in the corresponding vectors and the singular values to understand the functioning of that matrix. SVD can be used even if the matrix is not square, unlike Eigendecomposition (another form of matrix decay). The most useful feature of SVD is that we can use it to make a certain change of matrix inversion to non-square matrices. So we can say that SVD compresses Matrices.

SVD says that any matrix can be explained by the combination of these three things, or in other words any Matrix can be factorized as

$$A = U S V^T$$

which means that Matrix A can be defined as a Product of two Matrices U and V, and both U and V should be Orthogonal.

Orthogonal Matrices implies that $U^T U$ and $V^T V$ are both going to be, equal to the identity matrix, which is based on being linearly independent, from each other and all the V's are linearly independent to each other which means that if we multiply that matrix of use or matrix of these by its respective transpose we're going to get the identity matrix and S is a Diagonal Matrix Containing the singular Values.

U and V are derived from orthonormal eigenvectors chosen from AA^T and $A^T A$. S is a diagonal matrix with elements that are equal to root of the positive eigenvalues of AA^T or $A^T A$.

So for example if A is 100*10 matrix, that means we need a thousand numbers to represent this matrix. With SVD it is possible to represent it with fewer numbers, suppose that the Sigma (S)₁ = 3 Sigma (S)₂ = 2. The first two singular values are 3 and 2 respectively plus for all the other singular values, any greater than S₂ would be close to 0, it basically means that there's almost nothing in any of the other singular values in a matrix that means that if we were to write out the full form for A all of the S after S₂ has 0 contributions to the matrix A. And by using only two

singular values, summing the matrix form we will compress 1000 numbers inside the matrix to only 225.

In our use-case we have matrix of 43 arrays, each representing its PCA component. By using SVD we are compressing matrix A with 43 arrays to singular array with 43 items where each number is corresponding to the array of PCA component, thus representing 3358 numbers in Matrix A to 43 numbers.

```
array([[4228.79866993],
       [4239.19291201],
       [4229.95313186],
       [4229.94284536],
       [4188.58882426],
       [4188.58882426],
       [4233.29624474],
       [4247.55333532],
       [4247.55333532],
       [4233.29624474],
       [4336.55214322],
       [4336.76815975],
       [4336.76815975],
       [4337.41628932],
       [4279.86240138],
       [4278.25183487],
       [4338.28027542],
       [4274.21094204],
       [4274.18758891],
       [4272.38766132],
       [4273.63840699],
       [4273.61932922],
       [4278.01905382],
       [4278.58309697],
       [4272.36437688],
       [4340.44044066],
       [4271.73664862],
       [4278.61300851],
       [4271.19838728],
       [4267.96689684],
       [4269.79719901],
       [4341.52852328],
       [4274.82980019],
       [4273.21382452],
       [4272.335231],
       [4278.31168816],
       [4288.28814704],
       [4278.82311533],
       [4272.79807721],
       [4273.77534116],
       [4218.24709356],
       [4221.50291818],
       [4211.65138903]])
```

```
len(x)
```

```
43
```

Figure 3.8: SVD Components

and to check if we can inverse the SVD transformation and can get back the PCA components on the knowledge of our array dimensions.

```
[126] SVDInverse.svd.inverse_transform(x)
SVDInverse
array([[ 639.7241591, -406.29084192, 638.7337183, ..., -512.1430911,
        639.7241591, -585.21800459],
       [ 641.29619154, -487.51672323, 640.10371603, ..., -513.40252273,
        641.29619154, -586.43288152],
       [ 639.89888178, -406.43233804, 638.98810532, ..., -512.28350632,
        639.89888178, -585.34881799],
       ...,
       [ 638.12793784, -405.85840637, 637.1398186, ..., -518.86588405,
        638.12793784, -583.95082844],
       [ 638.62047124, -405.44060286, 637.6317619, ..., -511.24011229,
        638.62047124, -584.33987687],
       [ 637.13012293, -404.38435664, 636.14374092, ..., -518.86780871,
        637.13012293, -583.16212468]])
```

```
SVDInverse.reshape(78,43)
array([[ 639.7241591, -406.29084192, 638.7337183, ..., 215.82044337,
        -520.8671157, 217.88179952],
       [ -524.82882709, 219.73285567, -527.00811222, ..., -484.68050807,
        639.33948487, -479.64519731],
       [ 637.40797884, -408.75988451, 631.51070377, ..., 225.28718646,
        -534.25265489, 231.24436154],
       ...,
       [ -477.6436549, 211.4937187, -480.60081289, ..., -534.98037617,
        628.73127791, -529.85780817],
       [ 631.69844391, -525.19218284, 635.0543324, ..., 218.81175483,
        -581.18930867, 211.80016683],
       [ -586.12134069, 211.09057884, -518.86780871, ..., -518.86780871,
        637.13012293, -583.16212468]])
```

Figure 3.9: Inverse of SVD

After getting Dimensional reduced convex hull as a feature,the next feature that I created is Central Moments

Central Moments

The very old problem in Computer Vision is of, how do we recognize motion energy, motion history images is solved by Statistics. With the help of concept in Statistics called moments and using Image moments and Central moments, Computer could recognize the motion within the image

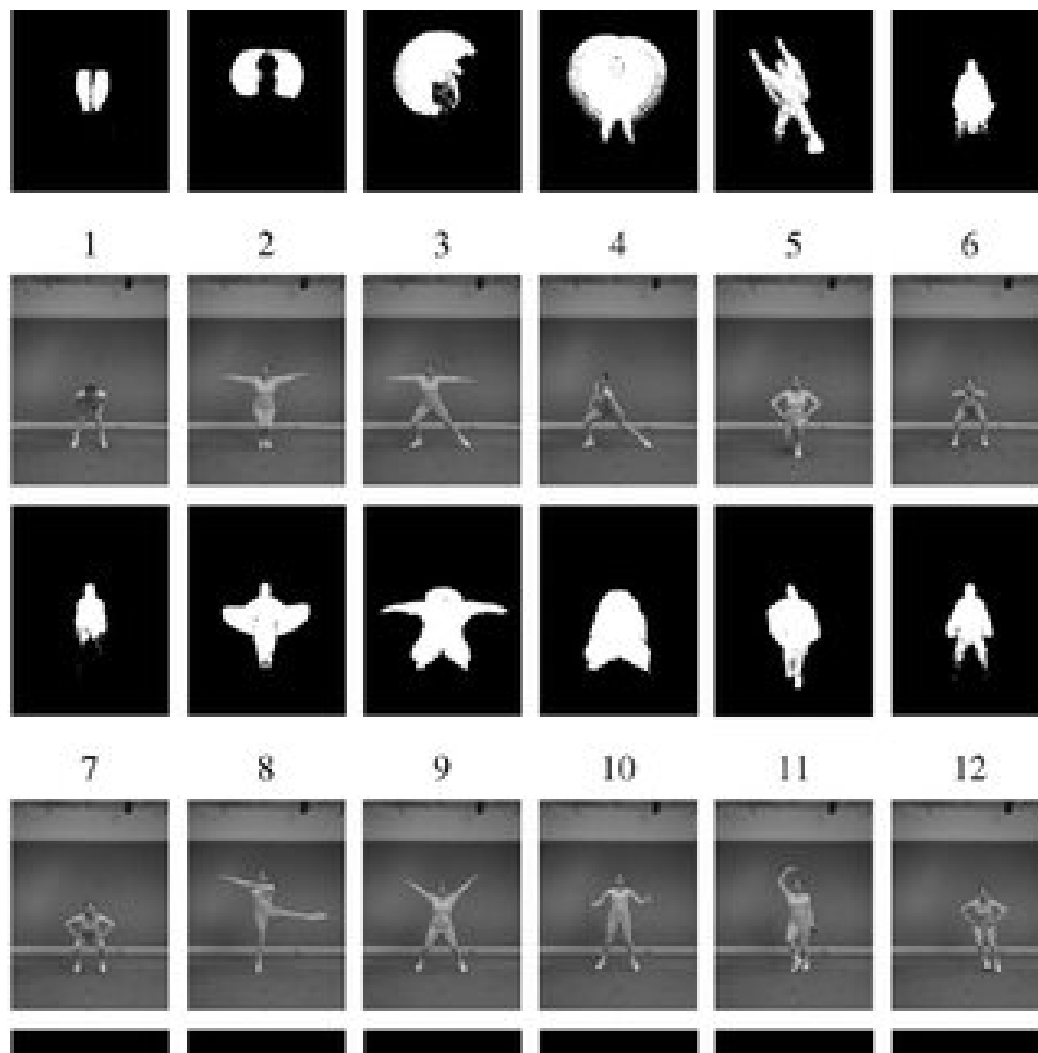


Figure 3.10: Davis and Bohick: The recognition of human movement using temporal templates

Moments

In both physics and statistics we are often interested in distances and this concept of

moments is actually shared between the two disciplines physics and statistics. The moment is the distribution of matter about a point or an axis. Moments can help us differentiate two datasets have similar distances with the help of different kind of moments.

Commonly used moments in Statistics

For a distribution, there could be a lot of moments. More moments will help in fully define the distribution. Though in Statistics, these four moments are commonly used

1 Mean(1st moment) It is the center of mass of the distribution.

2 Variance (2nd moment)- it is the degree to which the distribution of X is scattered.

3 Skewness (3rd moment)- It is a measure of the weight of a distribution in one direction or another. A normal distribution has no skew, positively skewed distribution have a low probability of extremely high outcomes, negatively skewed distributions have a small probability of extremely low outcomes.

4 Kurtosis (4th moment)-Kurtosis measures the extent to which X places more probability on the center of the distribution relative to the tails. Higher Kurtosis means less frequent larger deviations from the mean and more frequent smaller deviations.

Image moments and Central Moments

An Image moment is a number that describes the image. According to The concept of mean and variance, they are derived from first and second moments of the random variable. The moments of a random variable shows its distribution. Image moments are the same concept as moments. The first order moment will give the center of mass, where the pixel size by its intensity, the second order moment will show the variation of quantity between the weight. In the same way to find the frame inertia of an object, in our case movement of the hand, one can find it in Image moments.

But the problem with Image moments is that they are sensitive to the position of an object, but in our case, we are not sure of the exact position of the hand, so we need a

method which is not sensitive to position of the hand, So here central moments came in, which works with mean of the position of the object.

$$M_{ij} = \sum_{\substack{x \\ \text{Image Moment}}} \sum_{\substack{y \\ \text{Image Moment}}} x^i y^j I(x, y)$$

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy$$

Central Moments

In this problem, we already found contours, That are surrounding the hand and also highlighting position and the shape of the hand in that particular frame. After that I found Central Moments of the contour to find the difference in motion, Center of mass, inertia, density and shape of the hand in different frames. Now I have Central moments for each frame, For each frame I have four moments, so I have 43 arrays of 4 moments, and again we have a problem of meaningful dimension reduction

```

array([ 1.10439800e+06,  2.00111039e-11,  2.36183074e+05,  8.98031947e+04,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        -3.14321369e-09, -4.20252922e+05, -1.59791531e+05, -1.50631028e+05,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        1.99796033e+10, -3.49102036e+08,  4.14003691e+09,  1.49949619e+09,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        -1.81591357e+11, -4.79183908e+09, -4.06565418e+10, -1.64834848e+10,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00,  0.00000000e+00])

```

M01
M02
M03
M04

Figure 3.11: One of the 43 arrays with 4 moments

In this case PCA and SVD are not a good option because we are not working with variables and correlation, here we are working with different Moments, having their own character and meaning, plus each image is showing different blobs or moments or inertia/mass or density, and as explained before moments are based on distances, for example on an axis, a distance from zero to some point and deriving different moments basis on that. Here we have another useful Statistical Technique to solve this problem, which is Multi Dimensional Scaling (MDS)

Multi Dimensional Scaling (MDS)

Multi Dimensional Scaling is a way to minimize distances between different points, to help them visualize on 2d graph. It is the interpretation of distances or differences between sets of objects. These can be colors, faces, moments, or any kind of concept (Kruskal and Wish, 1978). Similar objects (or shorter distances) are closer together on the graph than very different objects (or longer distances). It is intended to represent high-Dimensional data in a small-scale area with the preservation of similarity between data points. This reduction in size is important in analyzing and revealing the actual structure hidden in the data. With clean data, reducing the size can effectively reduce the noise effect on the embedded structure. By setting big data, reducing the size can effectively reduce data retrieval. Therefore, MDS techniques are used in many data research programs and genetic network research.

When to Use MDS

Suppose you were given a list of city locations, and you were asked to create a map that included distances between cities. The process would be straightforward and would not involve anything as complicated as taking a ruler and measuring the distance between each city. However, what if you were given only the distances between cities (i.e. their similarities) - not their locations? You can still create a map - but it will include the right amount of geometry and other logical cuts. This type of logic problem is ideal for multidimensional scaling. You're basically given a set of different things, and the objective is to make a map that will tell you what the original distances were and where they were located.

But in this case of central moments, there are different moments in each frame, and MDS can find distance among moments of each frame and give unique X and Y based on similarity and dissimilarity. Which could hold a unique information of frame change in moments(inertia,weight,moment,pixel range of an object, in our case hand) and similarity of the moments in an sequence, which could be good classification feature for the sign languages objective,that is to put weights on unique features to connect them to word Cane.

```

[ 4.12780928e+11,  7.12280928e+11],
[-3.49180934e+11,  7.66652477e+10],
[-2.35690723e+11,  3.48886006e+10],
[-2.42302005e+11,  3.99158724e+10],
[-7.30416815e+10, -6.84039425e+10],
[-7.66302969e+10, -6.67711843e+10],
[-3.25989593e+11,  1.47934887e+11],
[-3.23043950e+11,  1.47952361e+11],
[ 1.02377055e+11,  7.39113223e+10],
[ 1.07957224e+11,  6.62886237e+10],
[ 5.31935207e+11, -3.56714518e+11],
[ 4.67539334e+11,  3.67923231e+10],
[ 3.61712388e+11, -3.31651465e+10],
[ 9.28503993e+10, -8.55881280e+08],
[-5.18934871e+10,  4.50546994e+10],
[-4.86255661e+10,  2.38683760e+10],
[-4.93353286e+10,  5.12028937e+10],
[ 2.53159305e+10,  2.15651993e+10],
[ 1.85680577e+10,  2.38158242e+10],
[ 7.22948732e+10, -6.46227817e+10],
[ 9.92098674e+10,  2.04427958e+10],
[ 9.34341488e+10,  2.55190013e+10],
[ 2.04691086e+10,  1.88508487e+10],
[-3.25870327e+10,  2.20488806e+10],
[-7.16879763e+09, -2.74658704e+10],
[ 1.11009129e+11,  2.42904019e+10],
[ 3.03624885e+11, -7.23870715e+10],
[ 2.12713273e+11, -1.14840964e+11],
[ 1.59225959e+11, -8.00058702e+10],
[-8.30598708e+10,  2.03607299e+10],
[ 9.94013942e+10, -4.13991336e+09],
[ 3.29895009e+10,  3.03838568e+10],
[ 1.45310550e+11, -1.21420843e+11],
[ 1.39173467e+11, -1.17245079e+11],
[ 2.15726026e+11, -1.66032791e+11],
[ 9.43606070e+10,  6.74687463e+10],

```

Figure 3.12: MDS 2d Points

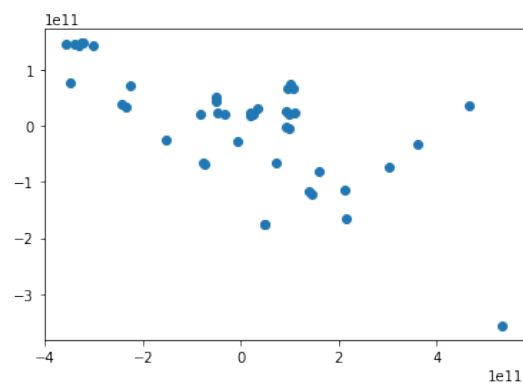


Figure 3.13: MDS points on Graph

Another feature I engineered is M Estimator for fitting line of the contour,

Fit Line

In open CV a line is fitted on basis of Robust Statistical method named M estimators which means minimizing the sum of squared error and residual and is resistant to outliers, it fits a line to 3D point set by minimizing distance between two extreme points of the contour that can be called Left side and right side. By finding extreme points of a fitline, for various frames, we can track the moment of the hand/ contour and in a way works as a good feature for position tracking for sequence of hand position in various different words



Figure 3.14: Fit Line in one of the frame of Word sequence

```
np.array([[leftline,rightline]..T
array([[ 770, 739],
[ 712, 744],
[ 624, 744],
[ 621, 745],
[-199, 760],
[-199, 760],
[ 776, 776],
[ 785, 785],
[ 785, 785],
[2380, 460],
[ 772, 772],
[1696, 736],
[1696, 736],
[ 774, 774],
[ 835, 732],
[ 841, 732],
[ 775, 775],
[ 898, 725],
[ 906, 724],
[ 844, 734],
[ 844, 728],
[ 841, 729],
[ 837, 730],
[ 855, 728],
[ 862, 729],
[ 772, 772],
[ 869, 727],
[ 878, 726],
[ 842, 730],
[ 855, 728],
[ 843, 729],
```

Figure 3.15: Left and right Extreme points array for each frame

Hence in this chapter I was able to generate three useful features that can be further used for a Sign Language detection Model, as I was able to extract useful information from sequence of frames and convert them into tabular data, which is storage efficient and can be used for multiple type of algorithms in Machine learning or Deep Learning for classification of the words, with various insights of Statistics methods used in Computer Vision and further reducing dimensions of features to a meaningful unique arrays with the help of Statistics Dimension reduction and scaling Techniques

Chapter 4

STATISTICS IN FINANCE-PROBLEM 3 STOCK FORECASTING

Stock market predictions are a long-standing problem that has been extensively analyzed using the tools and techniques of Machine learning and Data Science. Though investing in stock can have high profits or significant losses due to much volatility of stock prices. An ever learning and adopting stock prediction model may help to reduce risk and reveals potential return in Stock Trading. There is always a risk to investment in the Stock market due to its unpredictable behavior and by its character, the stock market is non-linear and volatile. Price Fluctuations in the series of the Stock market may depend on multiple factors, such as equity, interest rate, Securities, Options, and on various internal and external factors that are not easy to find and check. Stock markets have been engraved in the global economy so much that any fluctuation in the market influences our personal and corporate life and also the economic condition of the country.

In this problem of Stock Forecasting, I will use various well researched Statistical techniques for predicting Stocks, and also will discuss various techniques powered by Statistics for Model Selections and Data Collections.

Hidden Markov Model(Hmm)

Hidden Markov model (HMM) is a statistical signal forecasting model, which has been used to forecast various economical system and stock prices.

The hidden Markov model (HMM) is typically used to predict the hidden systems in the data. So, this model can be used in many areas, like speech recognition, molecular biology and financial market forecasts. HMM model has the following main assumptions:

1. an observation at any point was generated by a hidden state,
2. the hidden states are finite and satisfy the first-order Markov property,
3. the matrix of transition probabilities between these states is constant,
4. the observation at certain time of an HMM has a probability distribution corresponding with a possible hidden state.

A Markov chain is a model that tells us something about the probabilities of se-

quences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don't observe them directly.

A Hidden Markov model (HMM) allows us to talk about both observed events and hidden events that we think of as causal factors in our probabilistic model. The basic elements of Hidden Markov Model are:

- Length of observation data, T
- Number of states, N
- Number of symbols per state, M
- Observation sequence, $O = \{O_t, t = 1, 2, \dots, T\}$
- Hidden state sequence, $Q = \{q_t, t = 1, 2, \dots, T\}$
- Possible values of each state, $\{S_i, i = 1, 2, \dots, N\}$
- Possible symbols per state, $\{v_k, k = 1, 2, \dots, M\}$
- Transition matrix, $A = (a_{ij})$, where $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$, $i, j = 1, 2, \dots, N$
- Vector of initial probability of being in state (regime) S_i at time $t = 1$, $p = (p_i)$, where $p_i = P(q_1 = S_i)$, $i = 1, 2, \dots, N$
- Observation probability matrix, $B = (b_{ik})$, where $b_{ik} = P(O_t = v_k | q_t = S_i)$, $i = 1, 2, \dots, N$ and $k = 1, 2, \dots, M$.

We can better try to understand these basic elements using the figure below:

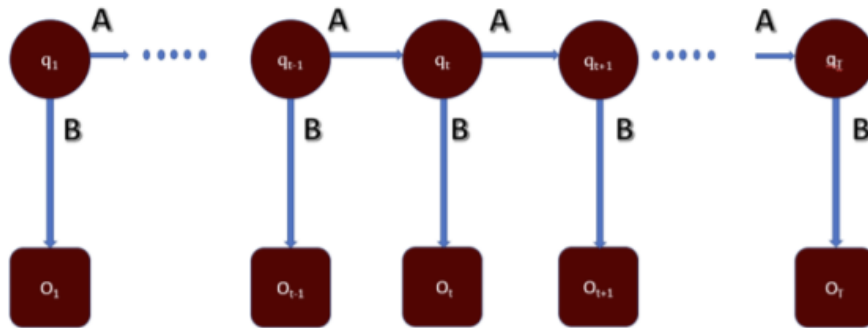


Figure 4.1: Basic elements of HMM

When working with Hidden Markov Model, we come across three fundamental problems:

1. Given the observation data and the model parameters, how do we compute the

probabilities of the observations?

2. Given the observation data and the model parameters, how do we choose the best corresponding state sequence?
3. Given the observation data, how do we calibrate HMM parameters?

Let's have an example I want to predict what is inside my Lunch at office today. I know according to my family history, I will eat one of three things for lunch at office: pasta, burrito, or dahl. My Mom does the cooking and she usually does not want to make the same thing two days in a row. However, what she cooks on one day is heavily influenced by what she cooked the previous day. Here's a graph that shows the probability of cooking one thing tomorrow given what she cooked today:

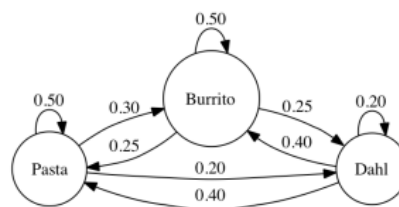


Figure 4.2: Probabilities For Cooking tomorrow, given what is cooked today

But there could be some additional factors that can help in improving the decision so, we can link those probabilities also. For example Packaging of the Lunch, is it in plastic Bowl or Tin foil.

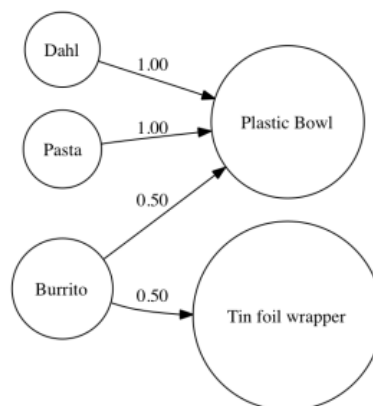


Figure 4.3: Probabilities For Cooking Given additional information

Once we've decided to model a process employing a Hidden Markov Model, there are three questions that usually come up:

The Evaluation Problem- for every possible sequence of observations (pasta, burritos, or dahl), what's the probability that the method will result in those observations? within the context of our example, for example, we'd ask
 What is the probability of finding a plastic bowl in my lunch bag on Monday, Tuesday, and Thursday, and a foil on Wednesday and Friday?
 How does this compare to the probability of finding a plastic bowl each day of the week? and same for foil.

what's the foremost likely sequence of Plastic bowl or foil wrapper, given cooking history?

The Decoding Problem- Given a sequence of observations, what sequence of states was presumably to guide to those observations?
 provided that a plastic bowl is inside my bag daily of the week, what was I possibly eating each day?

Given that a plastic bowl was found in my trash on Monday and Thursday, and a tin foil wrapper was found every other day, what was I presumably eating each day?

The Learning Problem Suppose we've some reason to expect that a selected sequence of observations is presumably to occur. How can we tweak our model parameters in order that the model predicts that this observation is most likely? This question comes up when we've decided to use a Hidden Markov Model, but we don't have already got perfect data about the possibilities involved within the choice.

The decisions one must make when designing HMM model are the following:

- 1 How many states do i believe my model needs?
- 2 How many possible observations do i believe there are?
- 3 What reasonably topology do i feel my state transition graph should have?

The first decision is typically made by someone who thinks they have already got a

bit little bit of insight into the method that's being modeled. If nobody is absolutely up to the current task, in practice people will make some different attempts at modeling and see what quantity of a difference it makes.

The first decision is sometimes made by someone who thinks they have already got a bit little bit of insight into the method that's being modeled. If nobody is basically up to the present task, in practice people will make some different attempts at modeling and see what proportion of a difference it makes.

The second decision is typically made by someone who has some insight into how the observations are made. If the observation is being made with a bit of apparatus that has very coarse resolution, there's no have to have very many possible observations within the model. The last decision is again one that needs some insight into the method. There are some common typologies that folks prefer to use as a start line, though. If you're unsure which to use, try them each and check the results. The second decision is sometimes made by someone who has some insight into how the observations are made. If the observation is being made with a chunk of apparatus that features a very coarse resolution, there's no have to have very many possible observations within the model.

The last decision is again one that needs some insight into the method. There are some common typologies that individuals wish to use as a place to begin, though. If you're unsure which to use, try them each and check the results.

Stock Market Forecasting with HMM

In stock market Forecasting, I am forecasting Closing price of the Banco BPM S.p.A. Stocks in Milano Stock Exchange using Guassian Hidden Markov Models.

HMM create combinations of the data we have and decide the probability of going from the current state to the next state. In order to forecast the Closing price of the stocks with HMM, it says that for example, we have three days' data, so the probability of the price on the third day given the price on the second day in the prices on the first day. We know given the Markov assumption in the hidden Markov model that the prices on the third day depend only on the prices on the second day and our goal is to maximize this probability because this would be maximizing the probability of the observed and hidden States and we want to find the combination of

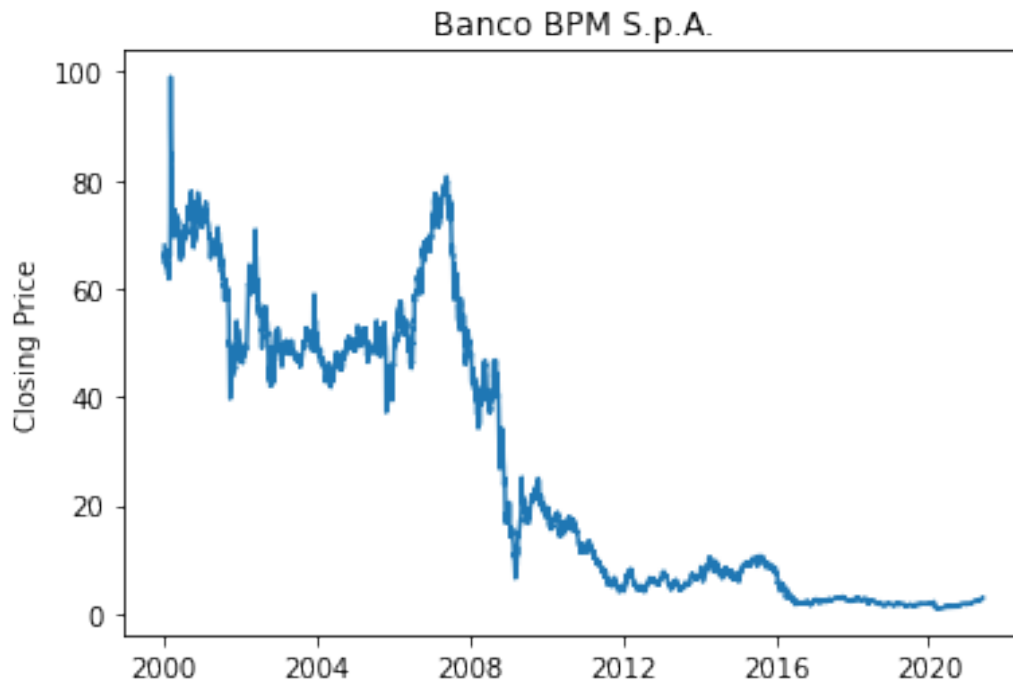


Figure 4.4: Banco BPM SPA Closing Stock prices

hidden States such that particular sequence is maximized. We know the transitions between the closing prices and some observed States like opening price, Volume and we can use all of this information to take some observed sequence of events, like change in closing prices, and maximize this probability of both seeing the observed data and seeing some combination of hidden states which we by finding all combinations of hidden States into a simplified product of probabilities based on Markov property of the hidden Markov model that we have assumed and once we find the maximizing hidden States we say that this combination of hidden States gives the most likely explanation for Seeing the given closing price.

Viterbi algorithm Viterbi algorithm is a programming algorithm finds the most probable sequence of hidden states given a sequence of observations in an HMM. It works by asking the question: given the parameter metrics and data, how to select states such that joint probability is maximized.

Observations I used following observations

Volume change = Volume change from previous day

1 Excessive High = Normalized High price of the day based on open ((Price High-open)/ Open)

2 Excessive Low = Normalized Low of the day price based on open ((Price Low-

open)/ Open)

3 Daily Change= Difference between Open and Close

4 Forecasted Change= Difference in open price from Day before

Model Selection in Time Series Statistics provide us some techniques to help in measuring different models with different Parameters for Time Series data. I use AIC and BIC for model Selection.

Log Likelihood

In simple terms log likelihood measures how strong the model is in fitting the data and the general rule is more complicated the model the better it fits the data.

K

But with log Likelihood, we measure the fitness of the model to data, but we also have a problem called overfitting overfitting which means using a more complicated model is going to fit the training data better but the more complicated you make your model the worse it's could be on the testing data basically. So k in AIC is a penalty term for complexity.

The Akaike information criterion (AIC)

AIC is a value that helps in comparing models as it includes measures of both how good a model fits the data and how complex the model is.

$$AIC = 2k - 2\ln(\hat{L})$$

AIC = Akaike information criterion

k = number of estimated parameters in the model

\hat{L} = maximum value of the likelihood function for the model

Figure 4.5: AIC formula

We pick a model with a low AIC which means that we're picking a model with a low k or a low number of parameters which is good we want to keep the model as simple as possible but we also need to make sure that the model has a high likeli-

hood. Low AIC helps in finding a model that fits the data well which means that the log-likelihood is high and the model relatively needs to have fewer parameters (k) and AIC helps in finding the model that has a balance between the two.

Bayesian information criterion (BIC)

$$BIC = n \ln(\sigma^2) + k \ln(n)$$

Figure 4.6: BIC formula

The difference between AIC and BIC is this coefficient in front of k , it's the natural log of the number of samples that we use in fitting the model. BIC takes also the number of samples used in training into account, We will pick the model with the lowest BIC because we want a model that fits the data really well so the likelihood is high and we want a model that uses relatively few parameters so k is low but the third part that wasn't present in AIC is that we also want to pick a model that was trained on the fewest number of samples.

For the problem of Stock Forecasting, the best selected model I got is,

```
model
↳ 100%|██████████| 6/6 [00:45<00:00, 7.64s/it]
   GaussianHMM(algorithm='viterbi', covariance_type='diag', covars_prior=0.01,
               covars_weight=1, init_params='stmc', means_prior=0, means_weight=0,
               min_covar=0.001, n_components=6, n_iter=10000, params='stmc',
               random_state=100, startprob_prior=1.0, tol=0.01, transmat_prior=1.0,
               verbose=False)
```

Figure 4.7: Best Selected Model

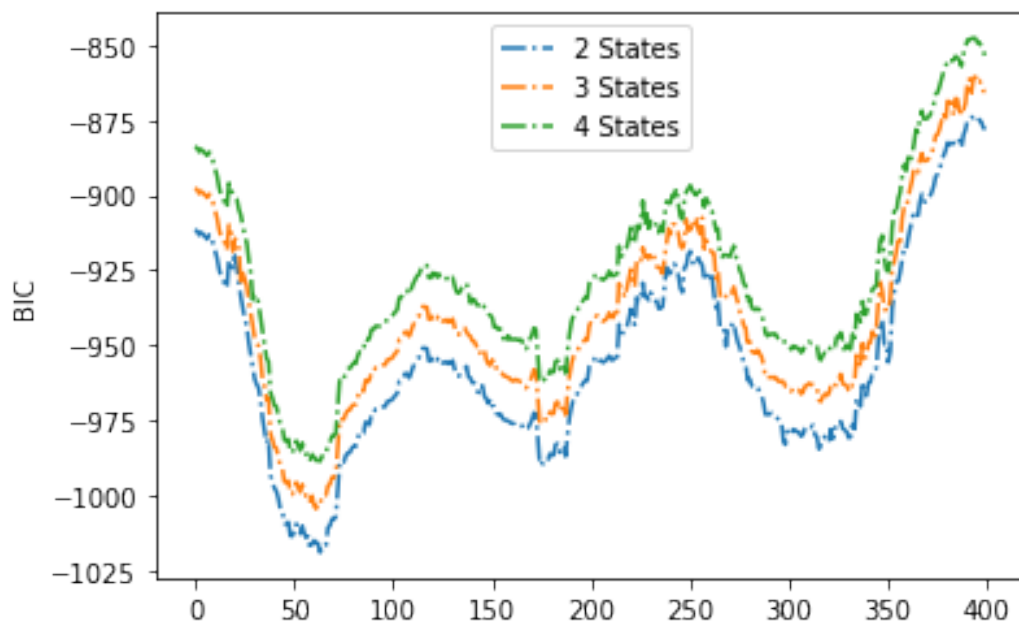


Figure 4.8: BIC to select number of states and best model

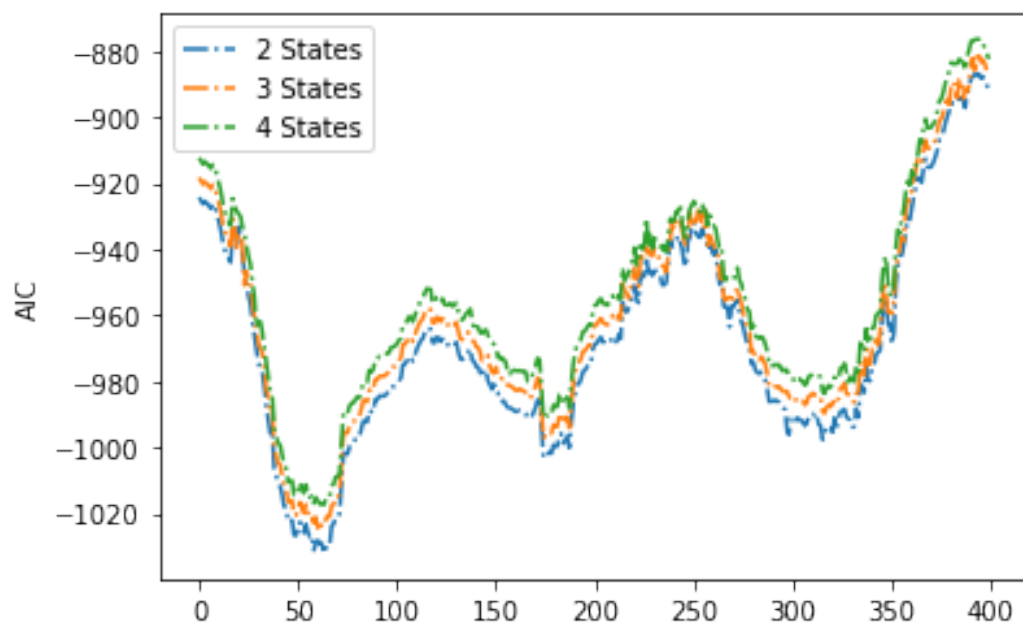


Figure 4.9: AIC to select number of states and best model

Variance Explained by different Hidden States of the best Model (Model for Banco BPM S.p.A)

The Colored lines indicates the mean and variance values of stock returns.

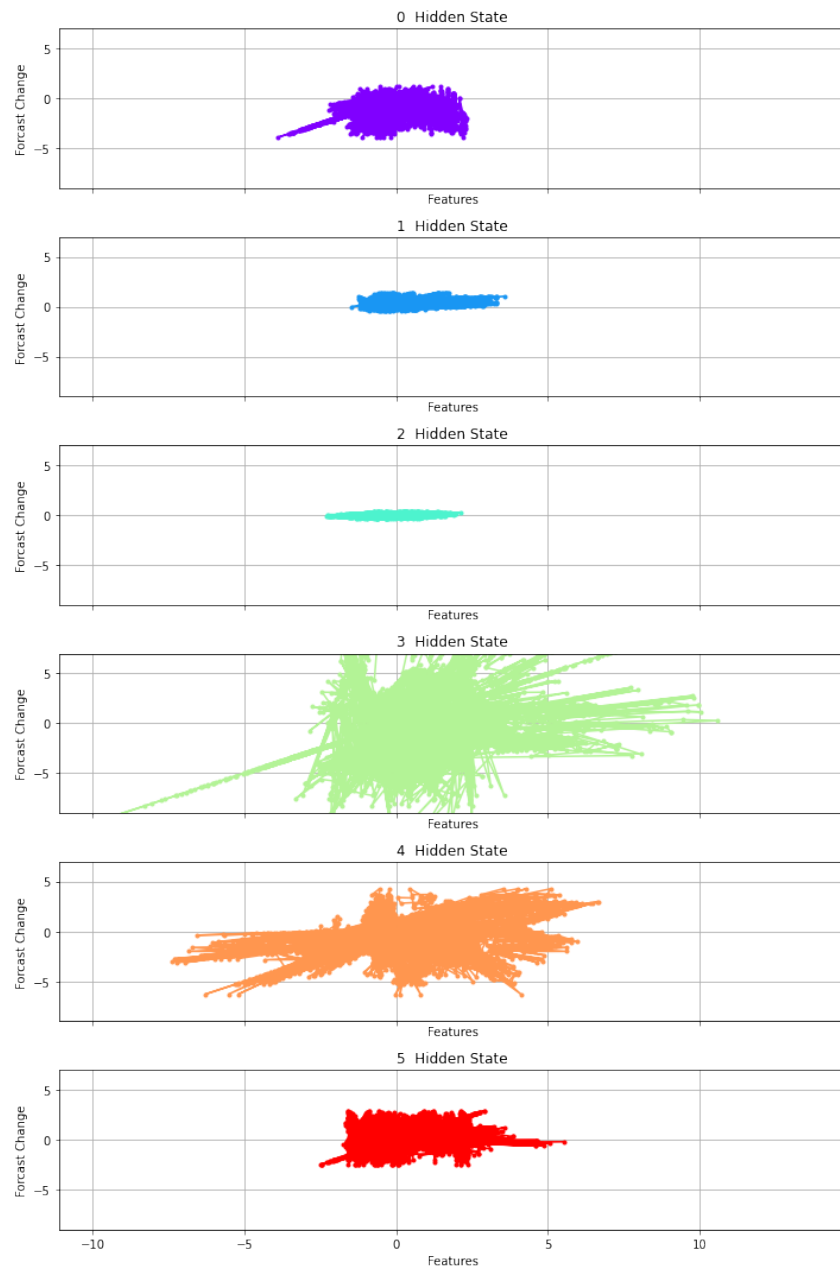


Figure 4.10: Variance Explained by different Hidden States of the best Model

We can interpret that the fourth hidden state represents the high volatility, based on the high variance, with negative returns. While the 0th and 1st hidden states

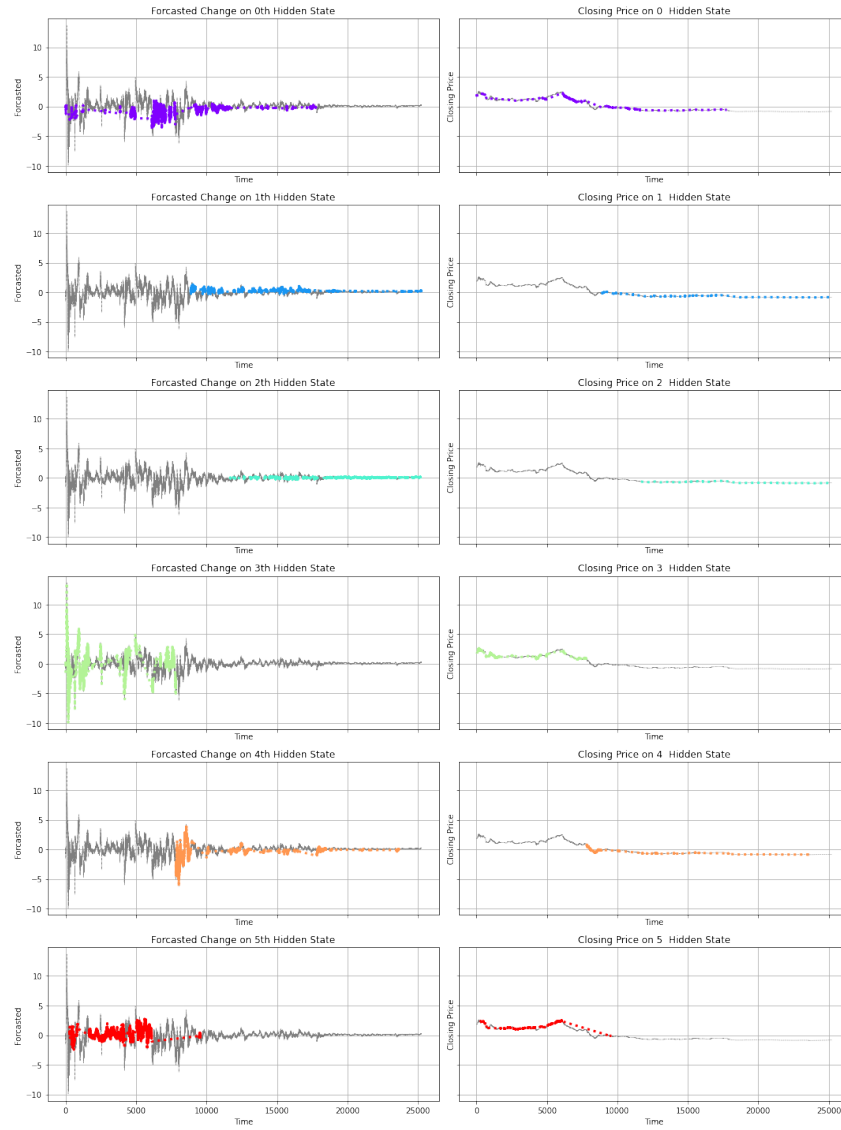
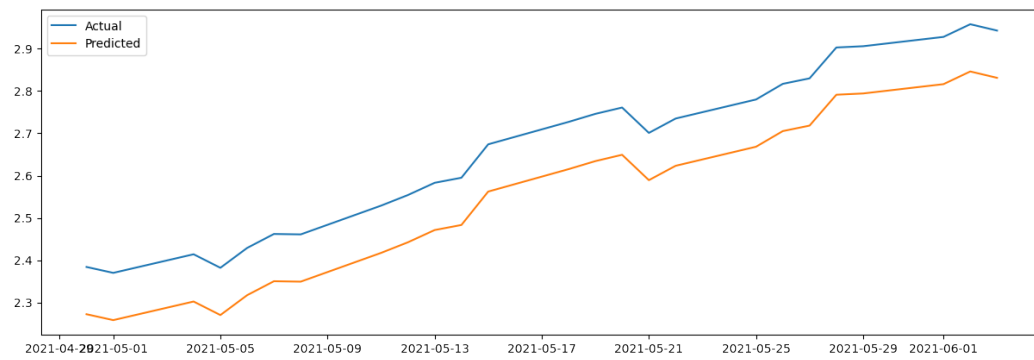


Figure 4.11: Different States explaining Volatility/Deviation in data

represent low and neutral volatility. This neutral volatility also shows the largest expected return

Stock Forecasting with HMM

I created a Self optimizing Algorithm with AIC and BIC and forecast last 25 days of the data some Stocks Closing Price with the same Algorithm and I get very interesting results.



Stock Forecasting with Seasonal Autoregressive Integrated Moving Average (SARIMA)

Autoregressive Integrated Moving Average ARIMA

For Time Series Forecasting on the basis of Auto Regressive and Moving Average, there is a rule that a time series need to be Stationary and should not have seasonality. ARIMA has some parameters that makes a time series Stationary, But lets understand Stationarity and try to remove manually Stationarity from our current use case data, Closing Stock Prices of Banca BPM SPA,

Stationary Stationarity means that the mean of the time series is constant, the standard deviation or Sigma of the time series is also constant over all time and there is no seasonality. It is an important concept as, many models assumes that the Time series we are working with is already Stationary, like AR, MR and ARMA models.

We can use various methods to make a Time Series Stationary, for example

- 1 Taking a Lag Difference,
- 2 shifting the mean
- 3 Taking Log of the Time series

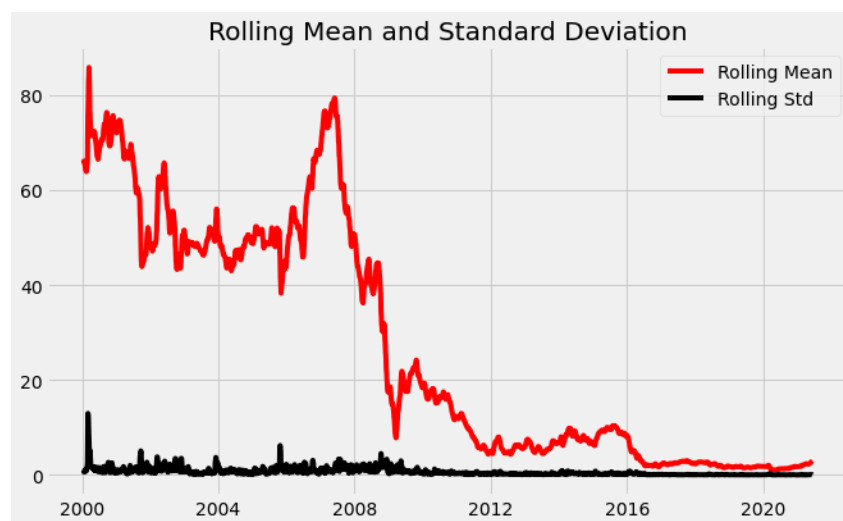


Figure 4.15: Mean and Standard Deviation of Closing Stock Prices of BANCA BPM SpA

As we can see that BANCA BPM SpA Closing Stock prices is not stationary, The

mean is not constant and also Standard deviation is not constant,

We do Augmented Dicky Fuller Test (ADF) to see if we were able to remove stationarity or not, if P-value is less then 0.05, means the series is Stationary. Here I differenced the time series with lag of 2 and then differenced the rolling mean of the lags to remove stationarity, to further remove Seasonality, I had to take log of the series.

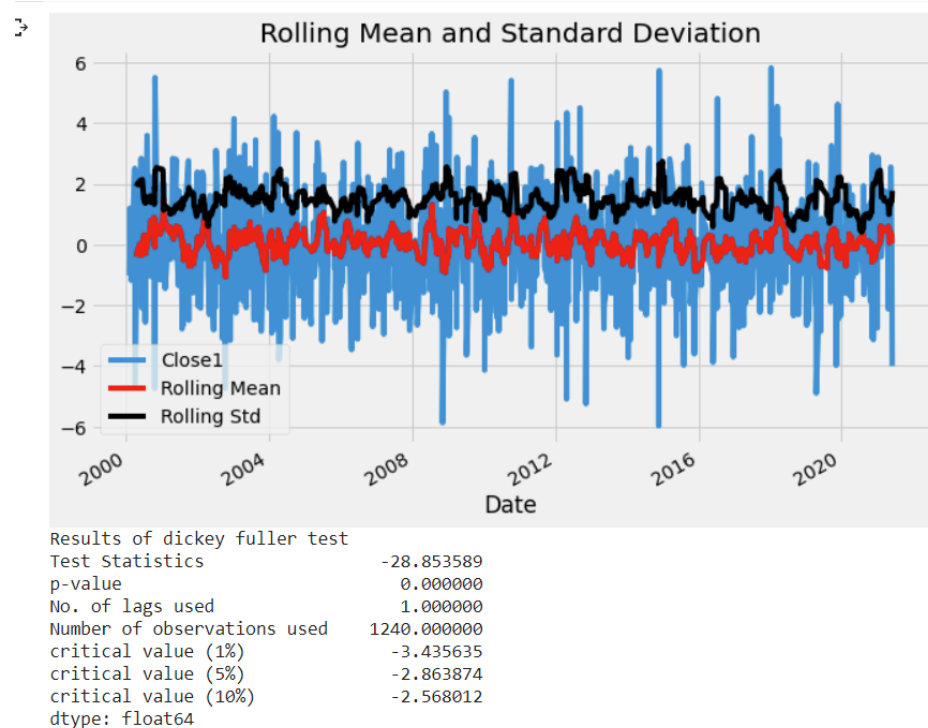


Figure 4.16: Manually Converted Time series to Stationary

P values of ADF test is 0.000 and visually also we can see that the series has constant mean and constant Standard deviation.

Now that is where ARIMA is different form ARMA model, ARIMA has an integration Feature.

Integration means doing a differencing, subtracting the previous value from each value, which makes the data more stationary.

Let us first discuss the major difference between AR and MA part for ARIMA,

Moving average(MA) model does not use the past forecast to predict the future instead it uses the residuals from the past forecasts. Whereas, the autoregressive model(AR) uses the past forecasts to predict future.

In order to remove Stationarity from the time series ARIMA uses following parameters,

d: Shows how many times the data needs to be differenced.

q: How many past data points to be used in MA part

p: How many past data points to be used in AR part

So for example, ARIMA(1, 3, 12) means combining the 1st order Auto-Regressive model and a 12th order Moving Average model. The 3 in between the 1 and the 12 represents the 'I' part of ARIMA(Integration) and it shows a model where taking the difference between response variable data to make mean and Standard deviation constant.

With integration part we are able to do have constant mean and Standard Deviation, but there is still one factor remaining to be considered, that is Seasonality. So here SARIMA, comes into action,

Seasonal Autoregressive Integrated Moving Average (SARIMA)

With SARIMA we add four new hyperparameters that will act on the Seasonality component of the Series plus an additional parameter for the period of seasonality.

P: Seasonal autoregressive order.

D: Seasonal difference order.

Q: Seasonal moving average order.

m: The number of time steps for a single seasonal period, Finding repetition of the Season

We select the best Model, by using AIC and BIC as we used in HMM model. Model with lowest AIC will be the best model.

```
ans = []
best_terms = (0,0)
best_aic = 99999 ## arbitrary number to replace
for comb in pdq:
    for combs in pdqs:
        try:
            mod = sarimax.SARIMAX(hist['Close'].values,
                                  order=comb,
                                  seasonal_order=combs,
                                  enforce_stationarity=False,
                                  enforce_invertibility=False)

            output = mod.fit()
            ans.append([comb, combs, output.aic])
            if output.aic < best_aic:
                best_aic = output.aic
                best_terms = (comb, combs)
        except:
            continue
    print(f'SARIMA {comb} x {combs} : AIC Calculated = {output.aic}')

... SARIMA (0, 1, 0) x (0, 1, 0, 12) : AIC Calculated = 15890.3681822927
SARIMA (0, 1, 0) x (0, 1, 1, 12) : AIC Calculated = 12031.574286407202
SARIMA (0, 1, 0) x (0, 1, 2, 12) : AIC Calculated = 11970.305336354902
SARIMA (0, 1, 0) x (0, 2, 0, 12) : AIC Calculated = 21785.635306373137
SARIMA (0, 1, 0) x (0, 2, 1, 12) : AIC Calculated = 15856.72947902725
SARIMA (0, 1, 0) x (0, 2, 2, 12) : AIC Calculated = 11732.921584126452
SARIMA (0, 1, 0) x (1, 1, 0, 12) : AIC Calculated = 14414.163466389873
SARIMA (0, 1, 0) x (1, 1, 1, 12) : AIC Calculated = 12037.15948291776
SARIMA (0, 1, 0) x (1, 1, 2, 12) : AIC Calculated = 11861.489492374167
SARIMA (0, 1, 0) x (1, 2, 0, 12) : AIC Calculated = 18777.465542402802
SARIMA (0, 1, 0) x (1, 2, 1, 12) : AIC Calculated = 14459.058977752633
```

Figure 4.17: Iterating to find Best Model with low AIC

Selecting Model on Bias Variance Trade off It is not necessary that the model with lowest AIC is the best, as we have to take a decision on the bases of the phenomenon of overfitting and underfitting. We need to Select a model, that do not have too much parameters and still have Low AIC

SARIMA MODEL fitted on Closing Stock prices of Banca BPM SPA

Statespace Model Results

Dep. Variable:	Close	No. Observations:	5480
Model:	SARIMAX(0, 1, 0)x(0, 2, 2, 12)	Log Likelihood	-5863.461
Date:	Sat, 12 Jun 2021	AIC	11732.922
Time:	14:27:20	BIC	11752.721
Sample:	0	HQIC	11739.832
	- 5480		

Covariance Type: opg

	coef	std err	z	P> z	[0.025 0.975]
ma.S.L12	-1.9860	0.007	-278.501	0.000	-2.000 -1.972
ma.S.L24	0.9861	0.007	140.412	0.000	0.972 1.000
sigma2	0.4935	0.004	116.327	0.000	0.485 0.502

Ljung-Box (Q): 174.57 Jarque-Bera (JB): 164868.48
 Prob(Q): 0.00 Prob(JB): 0.00
 Heteroskedasticity (H): 0.02 Skew: -1.25
 Prob(H) (two-sided): 0.00 Kurtosis: 29.88

Warnings:
 [1] Covariance matrix calculated using the outer product of gradients (complex-step).

Figure 4.18: SARIMA model Summary

SARIMA Results for Banco BPM SPA



Figure 4.19: SARIMA model Accuracy Results

SARIMA Forecast for Banco BPM SPA

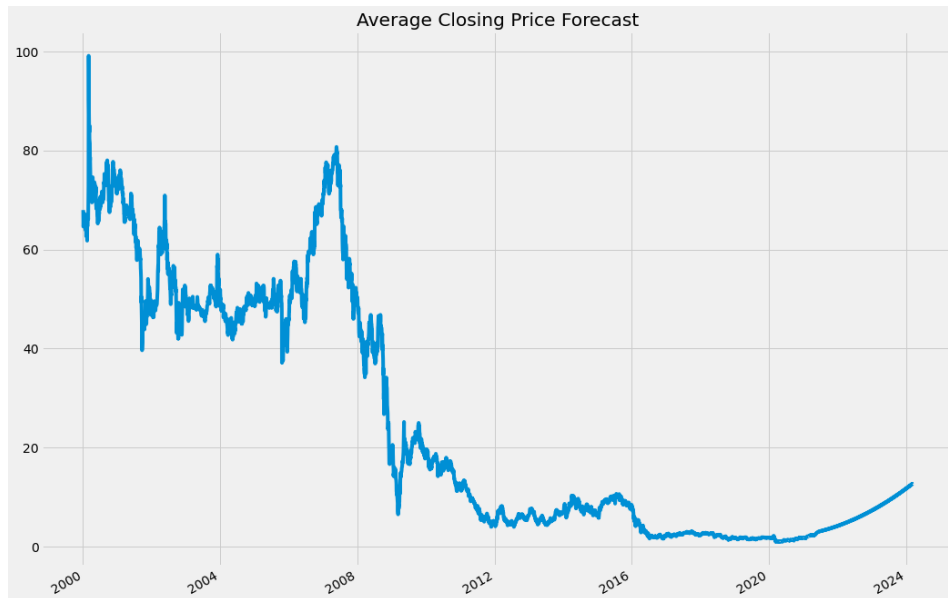


Figure 4.20: Sarima Model Forecast

Hidden Markov Model VS SARIMA

Statistics provide us various Evaluation Metrics, to compare the performance of different Models with each other. Some of them are

Mean Squared Error: In statistics, the mean squared error (MSE) of an estimator measures the average of the squares of the errors or in other words the sum of the difference between the original Closing price and the price predicted by the model. So lower the MSE, better are the predictions

MSE of HMM for Banco BPM SPA

```
In [41]: from sklearn.metrics import mean_squared_error

mse = mean_squared_error(Actual, predicted_prices)
print('MSE: %f' % mse)

MSE: 0.012484
```

Figure 4.21: MSE of HMM for Banco BPM SPA

MSE of SARIMA for Banco BPM SPA

```
[108] mse = mean_squared_error(hist['close'].values[4500:5471], res.predict(start=4500, end= 5470, dy
print('MSE: '+str(mse))

MSE: 0.1344988881704154
```

Figure 4.22: MSE of SARIMA for Banco BPM SPA

Root Mean Squared Error:RMSE is the standard deviation of error also know as error of Standard Deviation.It shows the scatter of the residual errors. It is always positive, and a lower value or closer to zero is ideal.

RMSE of HMM for Banco BPM SPA

```
In [42]: rmse = math.sqrt(mse)
print('RMSE: %f' % rmse)

RMSE: 0.111733
```

Figure 4.23: RMSE of HMM for Banco BPM SPA

RMSE of SARIMA for Banco BPM SPA

```
[109] rmse = math.sqrt(mse)
      print('RMSE: %f' % rmse)
```

```
RMSE: 0.366741
```

Figure 4.24: RMSE of SARIMA for Banco BPM SPA

So, in this Chapter I explored various Statistical method to predict Stock Markets, and focused on various Evaluation metrics powered by Statistics to check the performance of the models.

Chapter 5

STATISTICS AND ARTIFICIAL INTELLIGENCE OPTIMIZATION

Artificial Intelligence or Deep Learning models learn to map or put weight on a set of inputs to outputs. It is impossible to find the perfect weights for a Deep Learning Model, Instead, the learning is optimized and an algorithm could be used to find the possible weights of the model that can help in learning with minimum loss in information and avoiding Bias or over fitting.

A deep learning model is optimized using various algorithm powered by Mathematics and Statistics fundamentals. I will discuss various Optimizers and Activation function used for Deep Learning.

For Use Case, I took famous Titanic Database, and trained Neural Networks to check the performance of each optimizer on this particular well know dataset.

Why Optimizers are important?

Optimizers are required to minimize the loss function and error by finding the Global minima of a cost function. It minimize the cost function by finding the best value for weights to be learned and making a Model intelligent and useful.

Cost Function

A cost function is a way to find, how well a model performed, given various parameters and hyperparameters. For example, the cost function in Linear Regression can be the sum of least square.

Optimizers

1 Gradient Decent

When we train a simple Machine Learning model, we try to discover the values of coefficients that will give us the lowest cost, in other words, we will find parameters that will help us reduce the error at most.

Now let us think in another way, suppose we put an AI in the multidimensional plane which is in the shape of the mountain, as an example see the image , and we put AI in the random starting position, also know as Initialization.

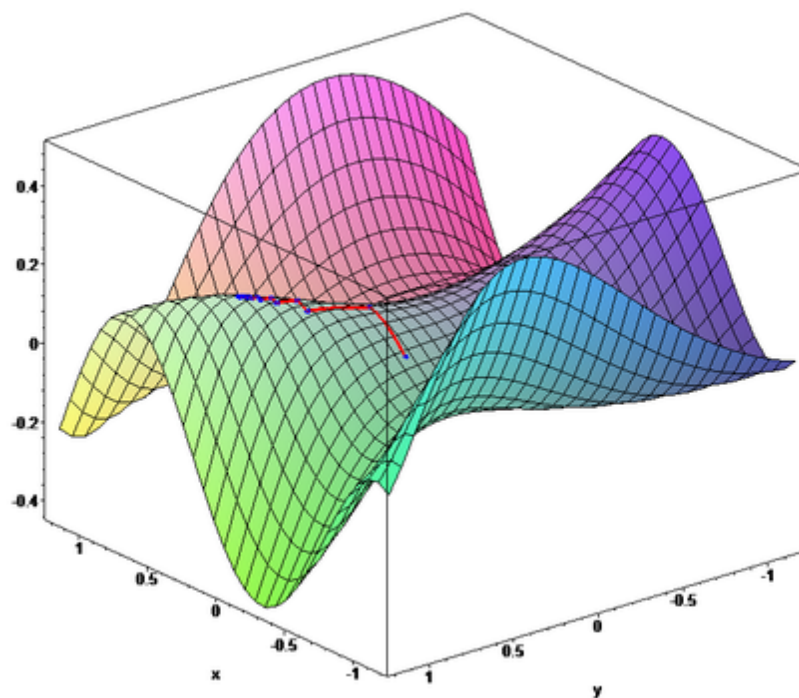


Figure 5.1: Multidimensional Plane of data, Where Ai is travelling to find Global Minima

In order to Learn, AI can only find only one thing that is Gradient, which is the point at which AI is standing, in technical terms rate of change of cost, given a change in coefficient. Gradient Descent will be achieved from the first-order differential of the point Gradient

Learning Rate : The learning rate can be understood as, how big a step AI will take

from going from the current point to the next point, or how many points it will skip, as AI is trying to find the lowest point by taking the first-order differential of each point and comparing it with previous. If the gradient is steep at the standing point and you take a large step, you will see a large decrease in the cost on the other hand if the gradient is small or closer to zero, then even if the large step is taken, the change in cost will be small, as we see many times training a deep learning model, even training more does not change the loss or accuracy rates.

So, in Gradient Descent, at every Gradient or point, AI will only know the parameter and the step size to take. And AI will keep on moving until it reaches “Convergence” and after enough steps, AI will be able to find the lowest point in the multidimensional plane. Convergence After enough steps, AI will realize that the cost is not improving by a lot of numbers and is kind of stuck which is also known as Global Minima, is the point of Convergence. The value of parameters at that last step is considered as the best parameters or coefficients for the model.

2 Stochastic Gradient Descent

There are a few drawbacks of the gradient descent algorithm. We need to check each and every iteration very closely. Suppose there are 10000 data points and 10 variables. Gradient descent will calculate the sum of squared residual for each data point and will calculate derivative with respect to each feature so 10000 multiplied by 10, computations will be needed to complete. So Stochastic Gradient Descent is an extension to Gradient descent, where Stochastic means Random. To calculate derivatives at each step, SGD will randomly select one data point from the whole dataset at every iteration. It will in a way sample a number of data points and create a mini-batch.

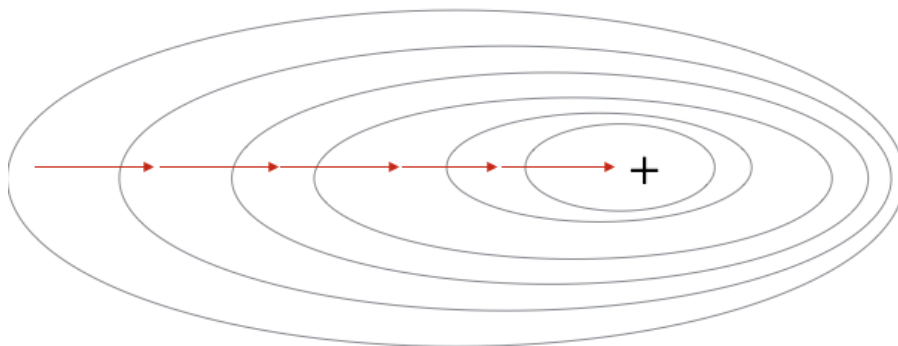
Advantages:

Memory requirement is less compared to the Gradient Descent

Disadvantages:

Each epoch will take more time to complete and also It Will take more time to converge, plus it can also get stuck in Local minima

Gradient Descent



Stochastic Gradient Descent

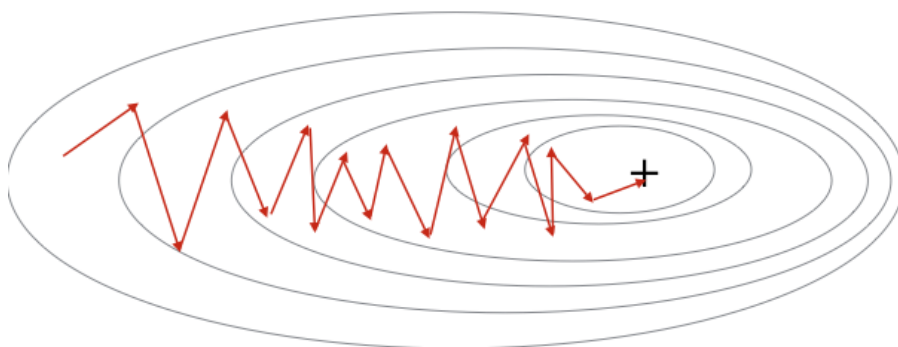


Figure 5.2: Gradient Descent Steps VS Stochastic GD steps

3 Stochastic Gradient Decent with Momentum

To solve the problem of time consumption with SDG, Momentum is introduced. Momentum is like a snowball going downhill which will gradually gain speed as it rolls down. So, in a way, as Algorithm will get closer to Global Minima, it will start taking large steps Momentum-based gradient descent will get in and out of the Minima valley as momentum increases and will need to come back to the same points more times before finally converging. In other words, Momentum will accelerate the convergence towards the correct direction and will reduce the fluctuations towards non-correct directions

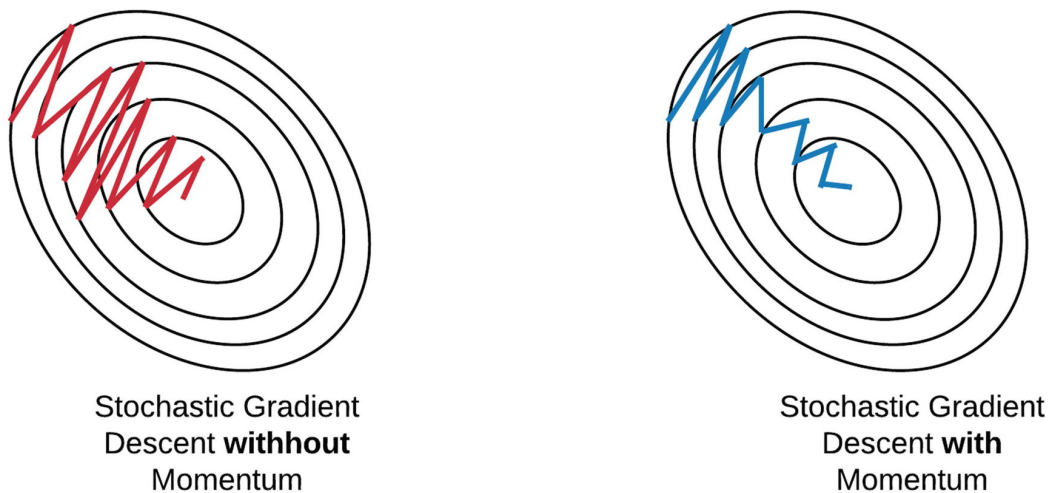


Figure 5.3: Stochastic Gradient Descent Steps VS Stochastic GD steps with Momentum

4 Nesterov Accelerated Gradient

Momentum can be a good alternative but it also has a drawback that it can miss a Local minima and may continue to accelerate. So NAG resolves this problem by looking into future location, so NAG calculates cost on the basis of future parameter instead of current.

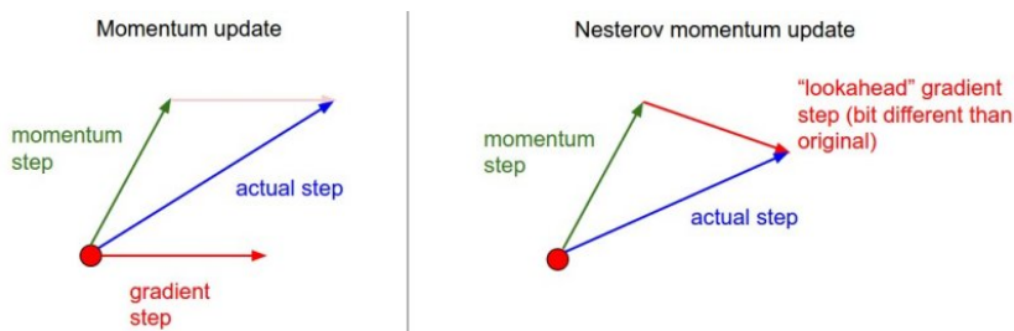


Figure 5.4: Momentum Steps VS NAG future Parameter

5 AdaGrad (Adaptive Gradient)

AdaGrad algorithm changes the learning rate and adapts the learning rate to the weights. It changes the learning rate for every weight and at each step. It makes a big step for less frequent weights and a small step for frequent weights. AdaGrad removes the need to tune the learning rate manually.

With a decreasing learning rate, it could be possible where the weights for common features converge faster to their Global minima, whereas for non frequent features, we may still need to observe more of them before determining Global minima.

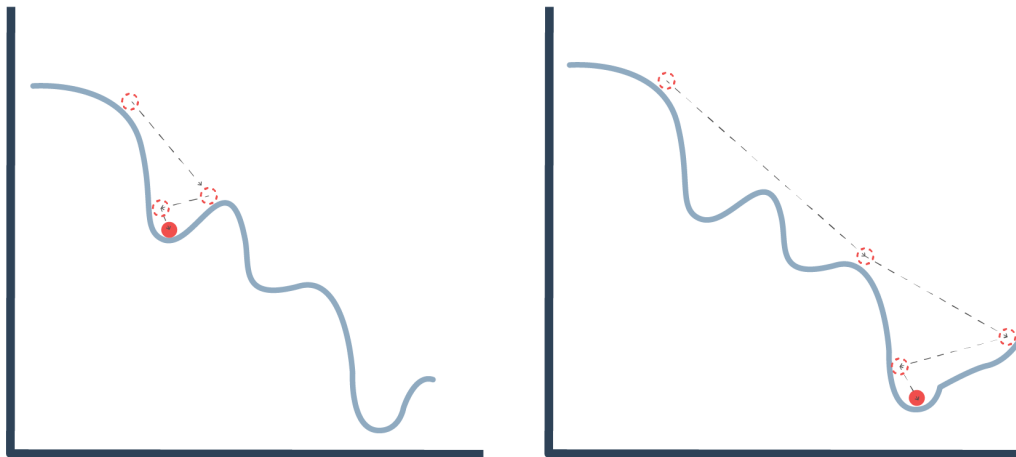


Figure 5.5: AdaGrad Learning Rate and Steps

5 AdaDelta

The drawback with AdaGrad is that the learning rate may become very small and iterations can be very large which will lead to slow convergence. To overcome this issue, the AdaDelta takes an exponentially moving average.

Exponentially Moving Average:

Exponential moving averages (EMAs) used to see trends of a value over certain time frames, Compared to moving averages, EMAs will give more weight to recent values than older values

So, in the place of summing all previously square gradients, Adadelata will put a limit of accumulation of gradients to a certain size. In other words, we are using the mean of the gradient to decay the learning rate.

5 ADAM (Adaptive Moment Estimation)

Adam Algorithm is the most recent and currently widely used Algorithm. It is a further improvement to Stochastic Gradient descent, Where Stochastic gradient

descent keeps the same learning rate (for all parameters, in Adam, it computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Though like in AdaGrade, in the place of adapting and changing learning rates based on the first moments, Adam also makes use of average second moments. The idea behind Adam is that we don't want to go very fast as we jump over the minima, so Adam decrease the speed for a bit thorough search,

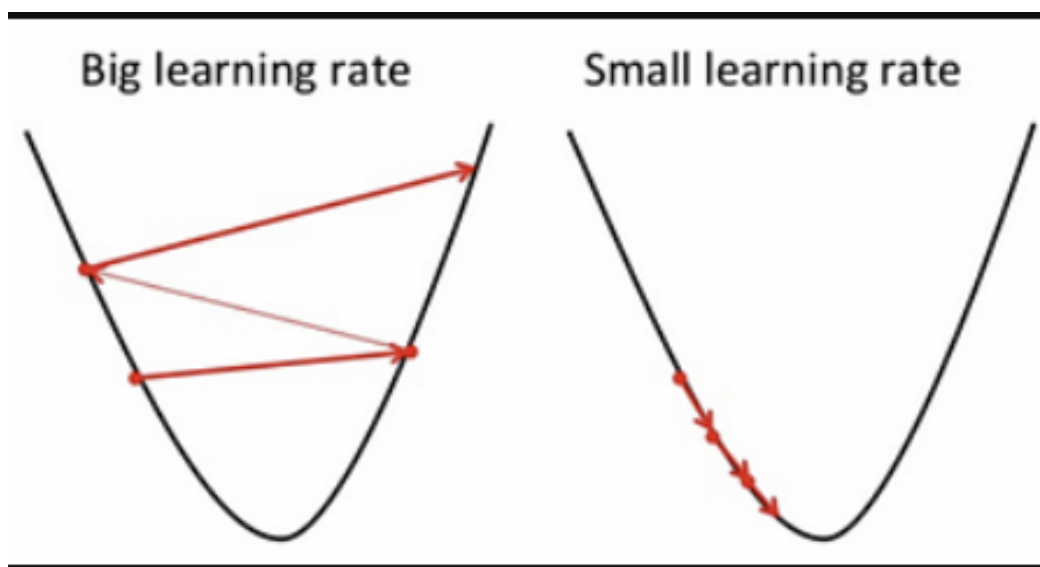


Figure 5.6: Adam Learning Rate

Validation and Training Loss over Titanic Dataset

I used popular Titanic Dataset to train Neural Networks using various optimizers, and to capture performance, though a performance of an optimizer hugely depends on the type of Data, So we can see SDG and Adam are more suitable for this Dataset.

ADAM Vs SDG

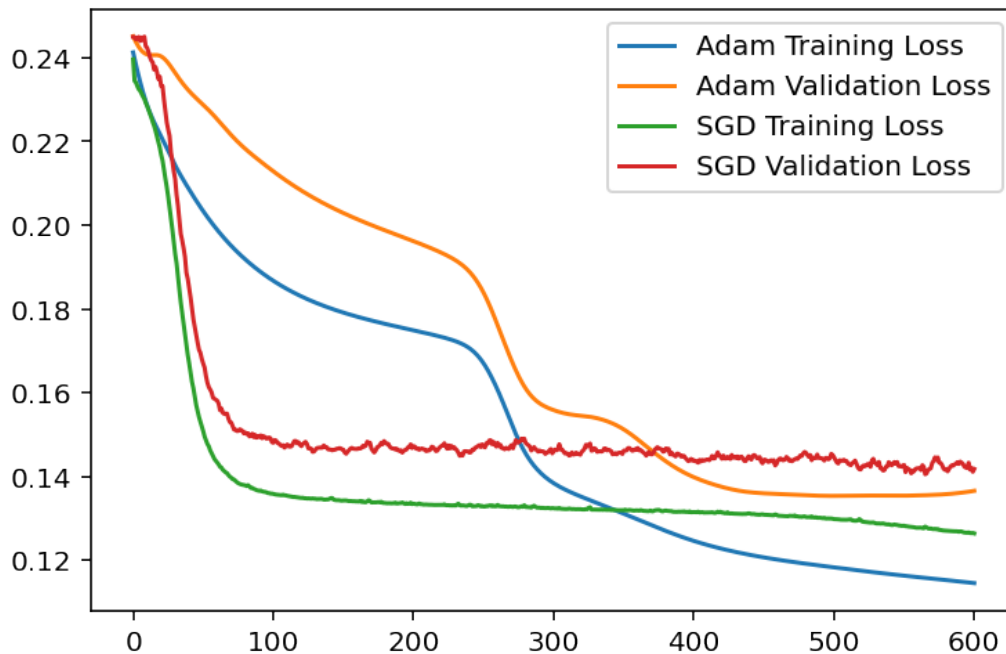


Figure 5.7: ADAM VS SDG over Titanic Dataset

AdaGrade Vs AdaDelta

From the plots, we can observed that using Adam, weights of the neural network are more smoothly adjusted to reduce the training loss. Try increasing the learning rate, and you can see that Adam converges much faster compared to SGD, using an adaptive learning rate.

The benefits of using Adam are not so obvious as the size of the data is very small and increasing training epochs tend to lead to overfitting and early-stopping is required. It is recommended to set the epochs for Adam to around 200 for the above hyperparameters configuration, as the training and validation loss starts diverging. However, we kept the epochs for both networks the same for plotting.

Lastly, in this implementation, Adam is much faster to compute compared to SGD

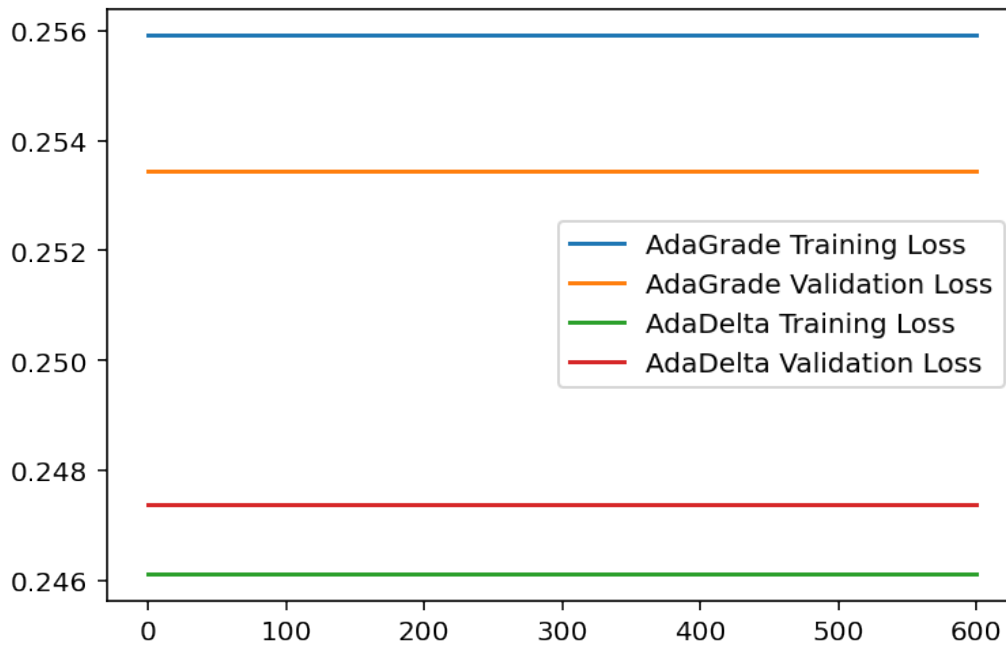


Figure 5.8: AdaGrade vs AdaDelta over Titanic Dataset

as it is processed as an entire training batch.

Model Accuracy over Test Data with different Optimizers

```
[43] print('Adam Test Accuracy: {}'.format(test_model(network_adam)))
      print('SGD Test Accuracy: {}'.format(test_model(network_sgd)))
      print('AdaGrad Test Accuracy: {}'.format(test_model(network_ada)))
      print('AdaDelta Test Accuracy: {}'.format(test_model(network_delta)))
```

```
Adam Test Accuracy: 0.8
SGD Test Accuracy: 0.8166666666666667
AdaGrad Test Accuracy: 0.4
AdaDelta Test Accuracy: 0.6
```

Figure 5.9: Model Accuracy over Test Data with different Optimizers

Activation Functions

We need Non Linear or Activation function to Classify complex data that we cannot put a straight line to classify, And also we need to select a line that do not compress/vanish the gradients.

Think of brain neurons, as some neurons fire up when signal has to be sent and others don't, Neurons connected to eyes, transpond the information from eyes and so on, Similarly Activation functions decide in neural networks, which neurons in the vast structure of our neural network created , needs to be activated.

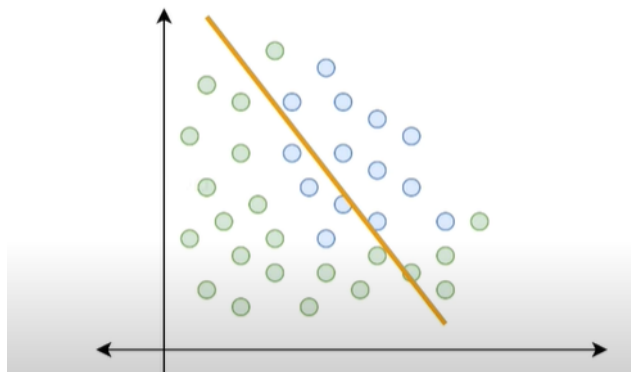


Figure 5.10: Linear function

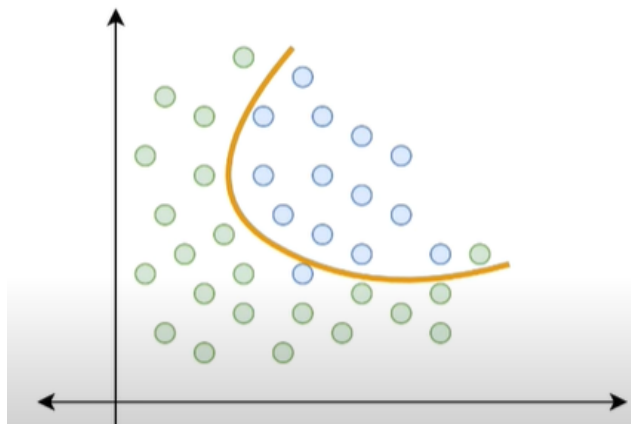


Figure 5.11: Non Linear function

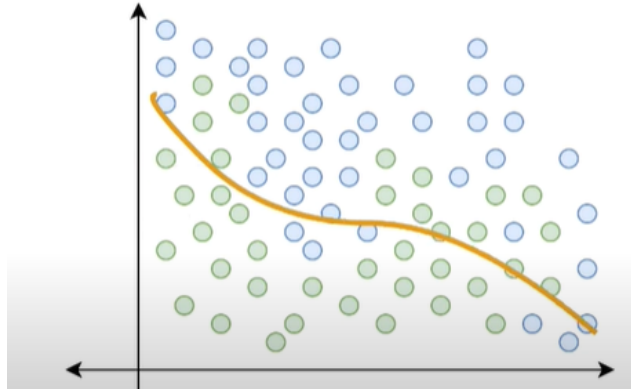


Figure 5.12: Sigmoid function

Commonly used Activation Function

There are vast number of activation functions and also we can also create our own custom Activation functions, though I will research and discuss some of the most commonly used activation Functions in Deep learning.

1 Step Function Step function looks like a step, which is based on a threshold, If a values is above threshold then its 1 else 0 It is very naive and also leads to information loss, as any value, way higher then threshold will still be considered same as the other values that have less influence

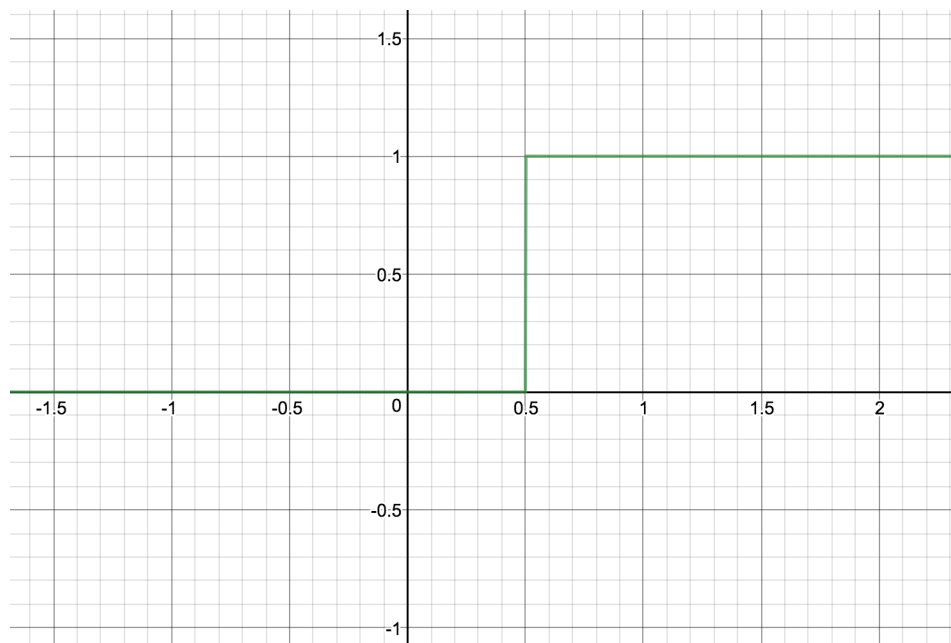


Figure 5.13: Step Function

2 Sigmoid Function Sigmoid function is a non linear function, which helps in capturing a spike near a boundaries so we can capture more influence near boundaries and when we are far from boundaries, we know it will have same influence on our decision, even how far it will go. The drawback is of vanishing effect, as after a certain point higher values are treated same.

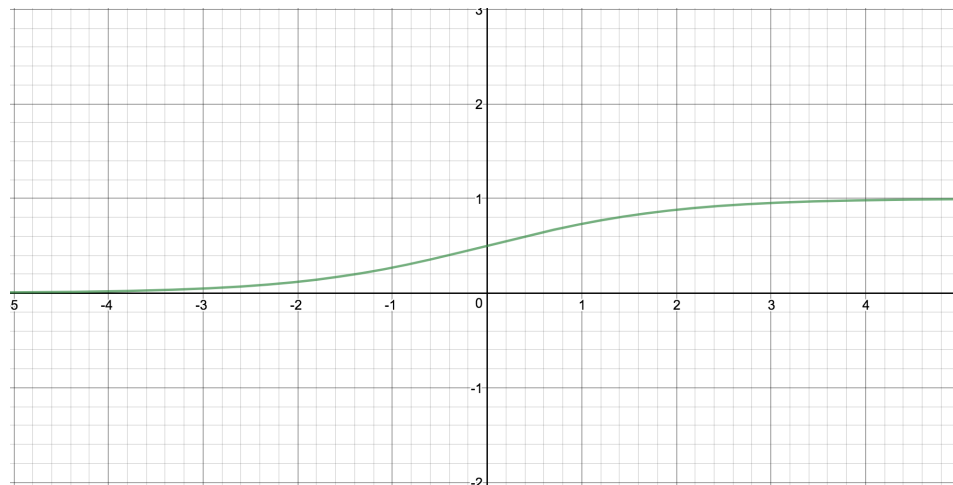


Figure 5.14: Sigmoid Function

3 Tanh Function Tanh is very similar to Sigmoid Function, The difference is that it is more steeper as Sigmoid function ranges between 0 and 1, while tanh function ranges between -1 to 1.

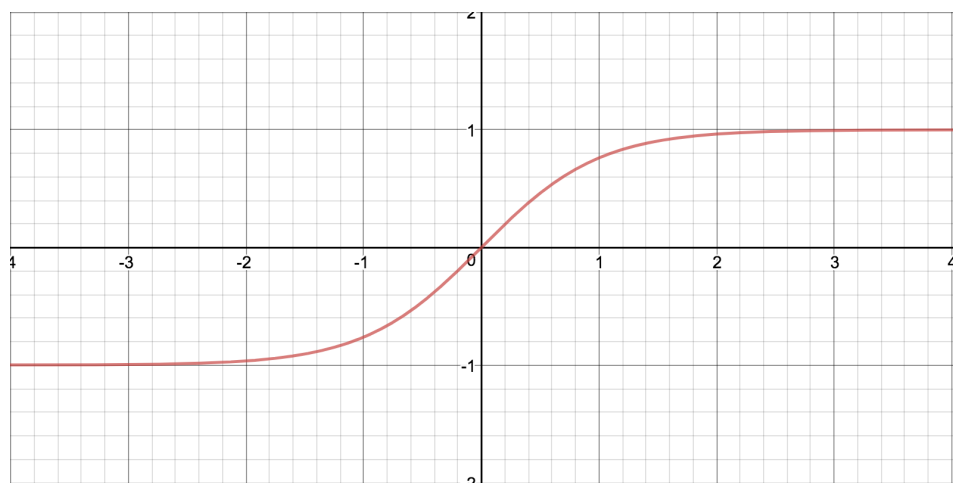


Figure 5.15: Tanh Function

4 ReLu ReLu gives output as the same number value if the value is positive otherwise it will give zero. ReLu looks like a Linear function, but it is nonlinear, and the range of ReLu is from zero to infinity, so it may also blow up the gradients. Suppose if we have a complex and vast structure of neural networks, using sigmoid or tanh will cause almost all neurons to activate. So with Relu almost 50 percent of the network will have zero activation as ReLu (output 0 for negative values of x

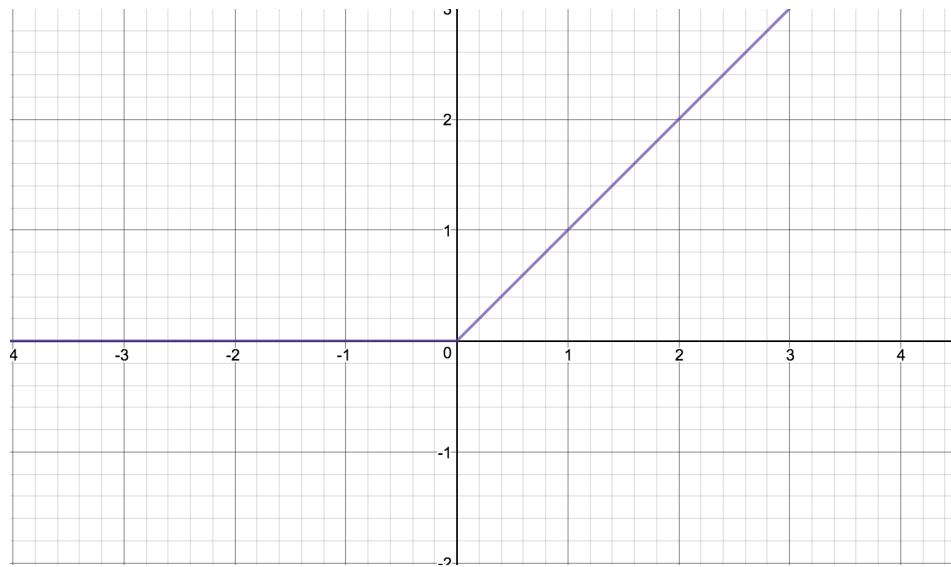


Figure 5.16: ReLu Function

5 Softmax Softmax is used in the final output layer. Softmax turns a vector of multiple values to the classification classes. The values in a vector can be in any range, the work of the softmax is to transfer them between 0 or 1, in other words into binary classifications. If an input is small, or negative, softmax will convert it into a small probability and if values are huge then it turns into big probability to classify into 0 or q

So, the performance of our network is based on Activation functions, For hidden layers, we need to select functions that we know, related to our particular problem will yield faster results and also will not have vanishing results with loss of information. For example, Sigmoid is always better when having Classification problems, though Relu is multipurpose and can be used when a network is very dense and performance is way low with any other network because they light up more neurons than Relu, Understanding the Activation function and shape, will lead to better building a neural network

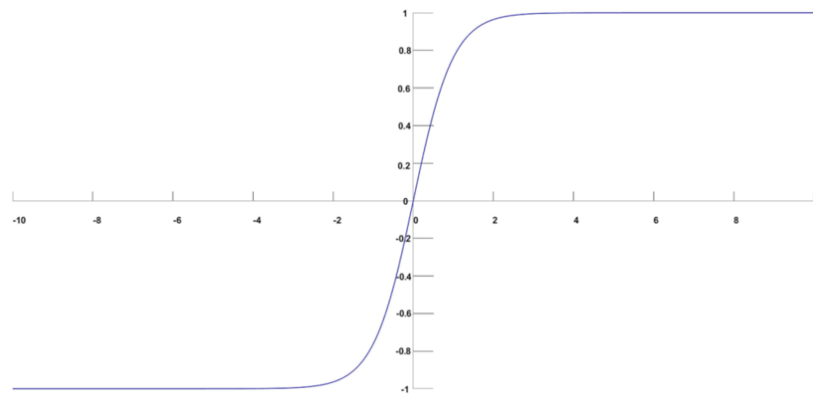


Figure 5.17: Softmax Function

Hence in this Chapter, I researched about various optimizers and Activation functions powered by the concepts of Statistics, that are very important to be considered while building a Deep Learning Model. I discussed about their Advantages, disadvantages and also implemented performance of optimizers over Titanic Dataset.

References and Citations

M. Su. Cluster Analysis: Chapter two Lecture notes, 2002,

S. Theodoridis and K. Koutroubas. Pattern Recognition. Academic Press, 1999.

J. Tou. DYNOC – A Dynamic Optimal Cluster-seeking Technique. International Journal of Computer and Information Sciences, vol. 8, no. 6, pp. 541-547, 1979.

R.H. Turi. Clustering-Based Colour Image Segmentation, PhD Thesis. Monash University, Australia, 2001.

P. Van Laarhoven and E. Aarts. Simulated Annealing: Theory and Applications.

C. Veenman, M. Reinders and E. Backer. A Maximum Variance Cluster Algorithm.

Image Segmentation. IEEE Transactions on Image Processing, vol. 12, no. 3, pp.

Mixture Models. IEEE Transactions of Pattern Analysis and Machine Intelligence, vol. 26, no. 5, pp. 651-656,

L. Gao, J. Song, X. Liu, J. Shao, J. Liu, J. Shao Learning in high-dimensional multimedia data: the state of the art Multimedia Syst., 23 (3) (2017), pp. 303-313

D. Amaratunga, J. Cabrera High-dimensional data J. Natl. Sci. Found., 44 (1) (2016), pp. 3-9

T. Ortner, P. Filzmoser, M. Rohm, C. Breiteneder, S. Brodinova Guided projections for analyzing the structure of high-dimensional data J. Comput. Graphical Stat. (2018), pp. 1-34

S. Bahrami, M. Shamsi A non-parametric approach for the activation detection of block design fmri simulated data using self-organizing maps and support vector machine J. Med. Signals Sens., 7 (3) (2017), p. 153

B. Tang, M. Shepherd, E. Milios, M.I. Heywood Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering International Workshop on Feature Selection for Data Mining, volume 39 (2005), pp. 81-88

Y. Tang, R. Rose A Study of Using Locality Preserving Projections for Feature Extraction in Speech Recognition IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP., IEEE (2008), pp. 1569-1572

C.-D. Chang, C.-C. Wang, B.C. Jiang Singular value decomposition based feature extraction technique for physiological signal analysis J. Med. Syst., 36 (3) (2012), pp. 1769-1777

S. Ng Principal component analysis to reduce dimension on digital image Procedia Comput. Sci., 111 (2017), pp. 113-119

P. Naik, M. Wedel, L. Bacon, A. Bodapati, E. Bradlow, W. Kamakura, J. Kreulen, P. Lenk, D.M. Madigan, A. Montgomery Challenges and opportunities in high-dimensional choice data analyses Mark. Lett., 19 (3–4) (2008), p. 201

L.V.D. Maaten, E. Postma, J.V. Herik Dimensionality reduction: a comparative

I.K. Fodor A survey of dimension reduction techniques, Lawrence Livermore National Lab., CA (US) (2002) Tech. rep.

Z. Zhang, F. Yang, K. Xia, R. Yang A supervised lpp algorithm and its application to face recognition [j] J. Electron. Inf. Technol., 3 (2008), p. 8

W. Liao, A. Pizurica, P. Scheunders, W. Philips, Y. Pi Semisupervised local discriminant analysis for feature extraction in hyperspectral images IEEE Trans. Geosci.

Remote Sens., 51 (1) (2013), pp. 184-198

X.L. Zhang, Nonlinear Dimensionality Reduction of Data by Deep Distributed Random Samplings, in: Asian Conference on Machine Learning, volume 2015, pp. 221–233.

A. Gisbrecht, B. Hammer Data visualization by nonlinear dimensionality reduction Wiley Interdiscip. Rev.: Data Min.Knowl. Discov., 5 (2) (2015), pp. 51-73

M. Verleysen, D. François The Curse of Dimensionality in Data Mining and Time Series Prediction International Work-Conference on Artificial Neural Networks, Springer (2005), pp. 758-770

On Image Analysis by The Methods of Moments Cho-Huak Teh and Roland T. Chin*

An Efficient Algorithm for Recognition of Human Actions Yaser Daanial Khan,Nabeel Sabir Khan,hoab Farooq,Adnan Abid,Sher Afzal Khan, Farooq Ahmad, and M. Khalid Mahmood

Stock Market Prediction Using Hidden Markov Models Aditya Gupta, Non-Student Member, IEEE and Bhuwan Dhingra, Non-Student member, IEEE

An Analysis and Implementation of the Hidden Markov Model to Technology Stock Prediction Nguyet Nguyen

Permanasari, Adhistya Hidayah, Indriana Bustoni, Isna Alfi. (2013). SARIMA (Seasonal ARIMA) implementation on time series to forecast the number of Malaria incidence. 203-207. 10.1109/ICITEED.2013.6676239.

Chen, Peng Niu, Aichen Liu, Duanyang Jiang, Wei Ma, Bin. (2018). Time Series Forecasting of Temperatures using SARIMA: An Example from Nanjing. IOP Conference Series: Materials Science and Engineering. 394. 052024. 10.1088/1757-

899X/394/5/052024.

A Survey of Optimization Methods from a Machine Learning Perspective Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao

Shaziya, Humera. (2020). A Study of the Optimization Algorithms in Deep Learning. 10.1109/ICISC44355.2019.9036442.

Activation Functions: Comparison of trends in Practice and Research for Deep Learning Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, Stephen Marshall

Andrew Ng Stanford University Verified email at cs.stanford.edu

ritvikmath of ritvikmath.com