# Capstone Project

# on

# Hand Gesture Recognition

**Submitted by**

**Group No. 5**


Moturu Praveen Bhargav -GBJALA8ZHP

Christo Thomas -9EHX3RNJUB

Farid Ahmed -TAZX6XUY0D

Sai Sandeep -LO44VDOBAM

VKS Pavithran -FTE9DUJGJH

Y V Surya -6C2O2GZQPVI


**Under the guidance of research supervisor**

**Ms Anjana Agrawal**

# INDUSTRY REVIEW

**Technology Industry:**

The technology sector is the category of stocks relating to the research, development and/or distribution of technologically based goods and services. This sector contains businesses revolving around the manufacturing of electronics, creation of software, computers or products and services relating to information technology. The technology sector offers a wide arrange of products and services for both customers and other businesses. Consumer goods like personal computers, mobile devices, wearable technology, home appliances, televisions and so on are continually being improved and sold to consumers with new features.

On the business side, companies are dependent on innovations coming out of the technology sector to create their enterprise software, manage their logistics systems, protect their databases, and generally provide the critical information and services that allow companies to make strategic business decisions. The term technology sector is frequently shortened to tech sector and is used interchangeably with the term technology industry. The technology sector is often the most attractive investment destination in any economy. The U.S. technology sector boasts of companies like Apple, Google, Amazon, Facebook, Netflix, IBM, and Microsoft. These companies drive the growth in the tech sector and the fervor around their long term potential has them trading at price-to-earnings multiples that look ridiculous compared to almost every other sector. A large amount of this growth owes a debt to the buzz factor that technology companies seem to effortlessly create by launching whole new business lines that have never existed before.

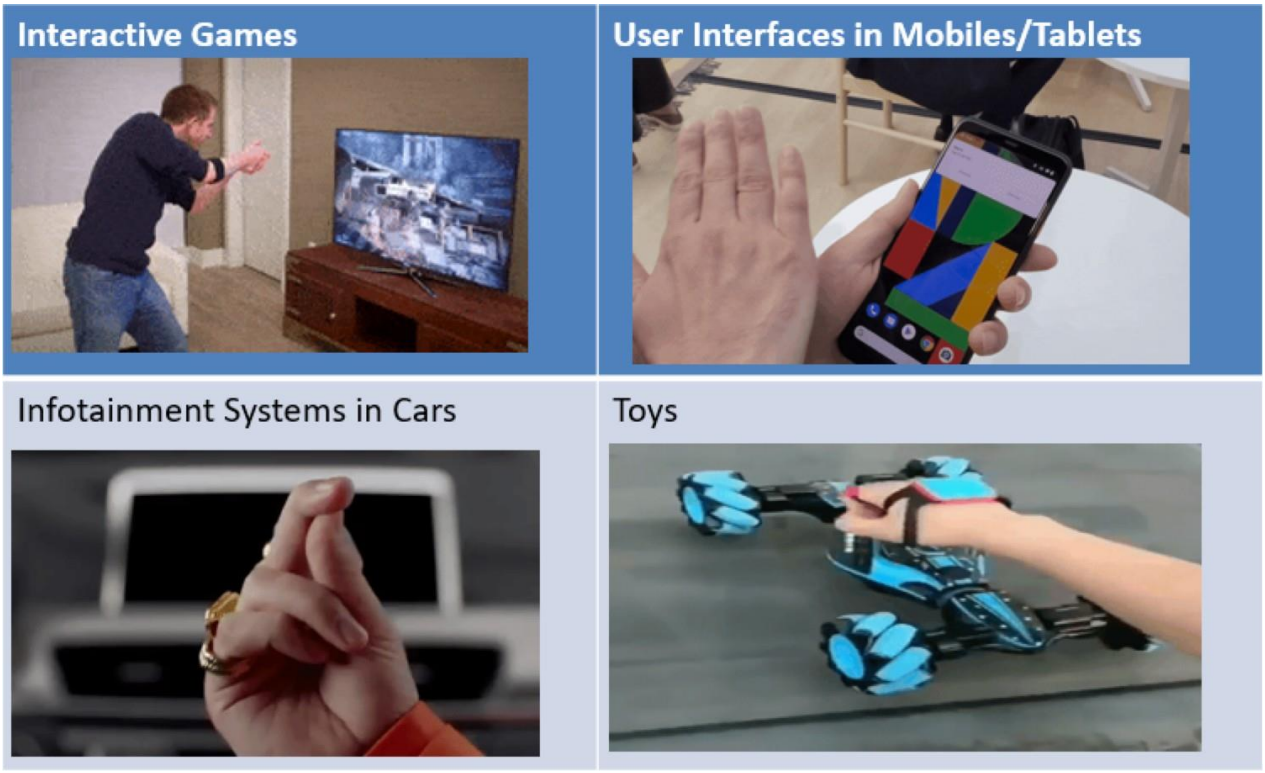
The term technology sector has been broadened many times to include businesses that may be better served by a more specific category. The technology sector was initially anchored in semiconductors, computing hardware, and communications equipment. The addition of software companies expanded the perceived tech sector to include anything based on coding. Soon, more room had to be made for Internet companies, which flooded during the Internet boom. Some of these Internet companies were media and content companies that just used code as the medium, but others were off launching rich features that grew to be e-commerce, social media, the sharing economy and even cloud-based computing. The technology sector now includes such a diverse set of companies that the subsectors are far more useful than the overall one. Unsurprisingly, there is no universal agreement—some pundits want a whole new sector for each innovation—but the big buckets include semiconductors, software, networking and Internet and hardware. From there, all the sub-sectors can be further broken down. For

example, hardware breaks into wearables, peripherals, laptops, desktops and so on. People may argue that it doesn't make sense to call a cloud computing company a software company, but the arbitrary separations are at least a bit more. The global technology industry is approximately worth $5 trillion as of 2019, manageable than the massive label of "tech sector" for every company.

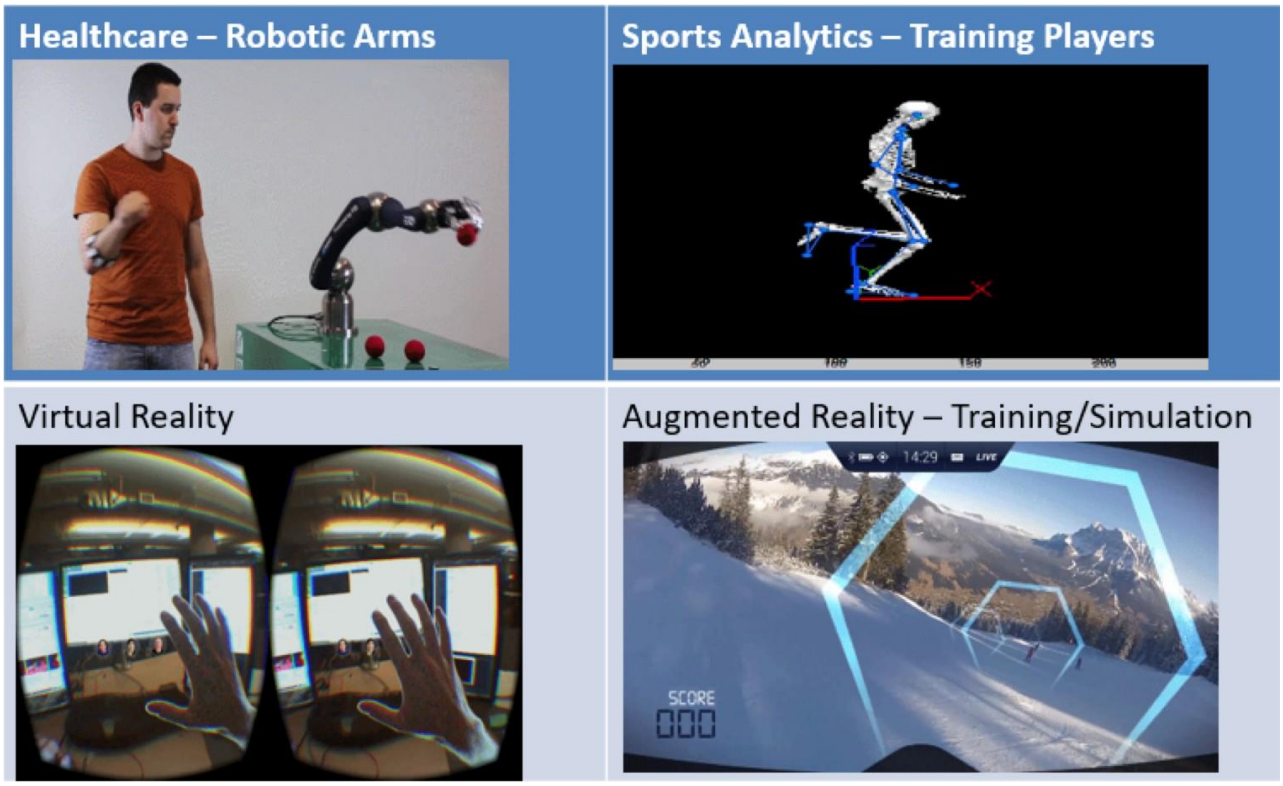
**Gesture Recognition**

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. Gestures can originate from any bodily motion or state but commonly originate from the face or hand. Users can use simple gestures to control or interact with devices without physically touching them. Gesture recognition can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse and interact naturally without any mechanical devices. Computer recognition of hand gestures may provide a more natural human-computer interface, allowing people to point, or rotate a CAD model by rotating their hands. Interactive computer games would be enhanced if the computer could understand players' hand gestures. Gesture recognition may even be useful to control household appliances. We distinguish two categories of gestures: static and dynamic. A static gesture is a particular hand conguration and pose, represented by a single image. A dynamic gesture is a moving gesture, represented by a sequence of images. We focus on the recognition of static gestures, although our method generalizes in a natural way to dynamic gestures. For the broadest possible application, a gesture recognition algorithm should be fast to compute. Here, we apply a simple pattern recognition method to hand gesture recognition, resulting in a fast, useable hand gesture recognition system.

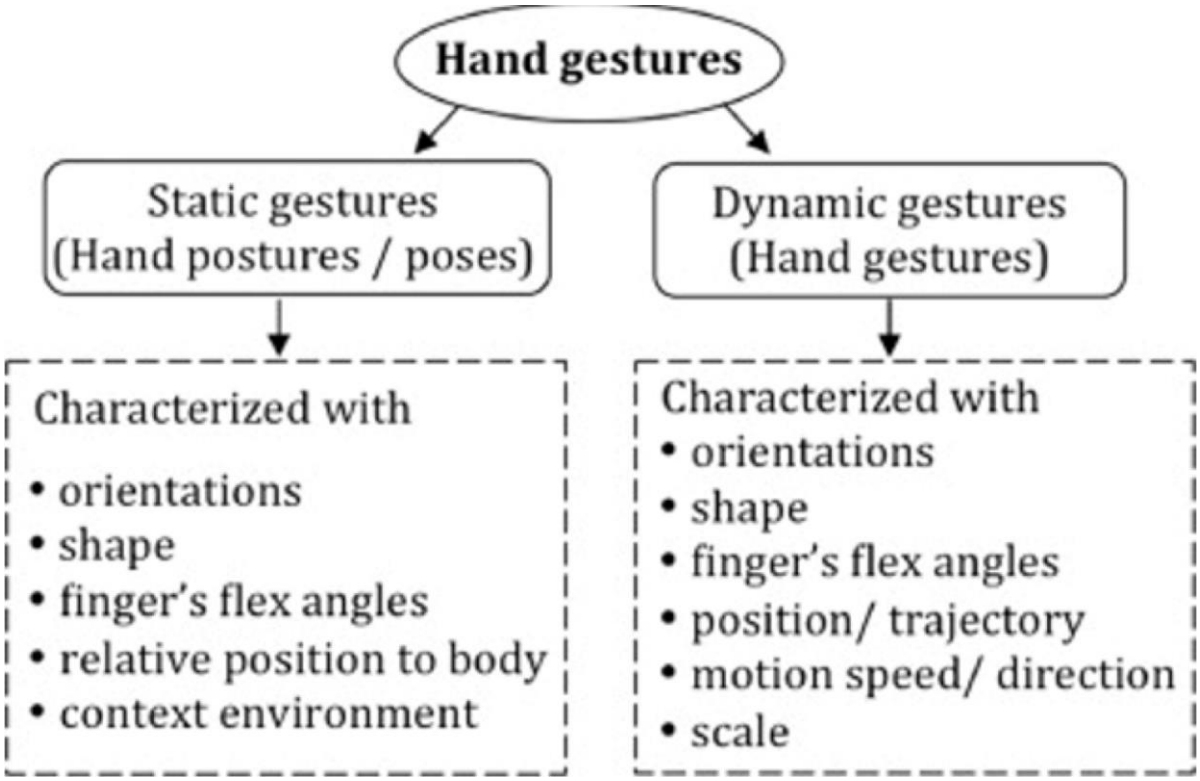
**Most Common Hand Gesture Control Use Cases:**




**Promising Hand Gesture Control Use Cases:**



**Hand Gesture Pointers**



**Gesture recognition market (2020 - 2025) :**

The Gesture Recognition Market is Segmented by Technology (Touch-based Gesture Recognition, Touch-less Gesture Recognition), End-user Industries like Automotive, Healthcare, Consumer Electronics, Gaming, Aerospace and Defence, and Other End-user Industries and Region.

**Market Overview**

The gesture recognition market is expected to register a CAGR of over 27.9% during the forecast period, 2020-2025. The development of artificial intelligence (AI) has given rise to gesture-recognition-based devices. Douwe Egbert has come up with an innovative machine, which was placed at the Tambe International Airport, which can detect travellers who yawned or looked sleepy and dispense free cups of coffee to them. The company was able to take benefit of face recognition technology to advertise and market its brand innovatively.

In the past, the interaction between humans and electronic devices was quite different. For instance, human interaction with TVs required a remote. However, today, gesture recognition technology is being increasingly implemented for human-device interaction, due to an increased acceptance of gesture-enabled electronic devices, across various industry verticals; for example, switching through television channels or radio stations.

The evolution of GUI technology from the use of texts as inputs to the use of gestures as inputs has paved the way for the emergence of gesture recognition technology. The use of gesture recognition is increasing in various sectors. One recent development in this area is the interaction of humans with machines, by using hand gesture recognition. Another development is the use of hand gesture recognition to control computer applications.With continuous technological developments, the companies in the market studied have been manufacturing products incorporated with new and innovative features. Omron Corporation has developed the gesture recognition technology, by simultaneously recognising the position, shape, and motion of a person's hand or finger, by referencing a camera-recorded image.

A gesture recognition application system comprises several key hardware and software components, all of which must be tightly integrated to provide a compelling user experience. A camera is the first component, which captures the raw data that represent the user’s actions. Generally, this raw data is then processed to reduce the noise in the signal, for example, or (in the case of 3D cameras) to compute the depth map. Moreover, specialized algorithms subsequently interpret the processed data, translating the movements into actionable commands that a computer can understand. Subsequently, an application integrates these actionable commands with user feedback, which must

be natural, as well as engaging. Adding to the overall complexity of the solution, the algorithms and applications are increasingly being implemented on embedded systems, with limited processing, storage, and other resources.

Adequate integration of these components, in order to deliver a compelling gesture control experience, is not a simple task. The complexity is further magnified by the demands of gesture control applications. In particular, gesture control systems must be highly interactive and be able to process significant volumes of data, with imperceptible latency.

**North America is Expected to Have a Major Share:**

The North American market for gesture recognition is led by the United States, due to the presence of major tech firms and startups in the country. R&D investment in the United States is very high. The country produces the most advanced degrees in science and engineering and high-impact scientific publications. It is the largest provider of information services, globally.

Deep learning forms a base for gesture recognition. In 2017, the deep-learning software market in the region was estimated to be USD 80 million and is expected to reach USD 130 million in 2019.

Also, in terms of artificial intelligence (taxonomy includes gesture-recognition-based products and services providers), the United States occupies the leading position with 415 companies, followed by the United Kingdom with 67 companies, and Canada with 29 companies. The average funding raised by the companies, particularly in the field of gesture control, is USD 7.8 million.

Canada-based Thalmic Labs manufactured a gesture recognition device that can be worn on the forearm, called Myo. This armband can be integrated with various applications, such as presentations and gaming or as a controller for drones. In terms of demand, the United States is helping in setting the stage for record sales of the latest consumer electronics. Disposable personal income increased by 1.8% in 2017 and is likely to increase by more than 2.0% in 2018. As a result, revenue in the consumer electronics industry is expected to amount to USD 72,443 million in 2018, in the United States.

**Competitive Landscape:**

The gesture recognition market is highly competitive and consists of several major players. In terms of market share, few of the major players currently dominate the market. These major players with a prominent share in the market are focusing on expanding their customer bases across foreign countries. These companies are leveraging on strategic collaborative initiatives to increase their market shares and profitability.

Gesture Recognition Market - Growth Rate by Region (2019-2024)

# LITERATURE REVIEW

**Rafiqul Zaman Khan and Noor Adnan Ibraheem (July 2012 ):**

In the paper titled HAND GESTURE RECOGNITION the author explained that the hand gesture recognition system received great attention in the recent few years because of its manifoldness applications and the ability to interact with machine efficiently through human computer interaction. In this paper a survey of recent hand gesture recognition systems is presented. Key issues of hand gesture recognition system are presented with challenges of gesture system. Review methods of recent postures and gestures recognition system presented as well .various methods are discussed for gesture recognition, these methods include from Neural Network, HMM, fuzzy c-means clustering, besides using orientation histogram for features representation. For dynamic gestures HMM tools are perfect and have shown its efficiency especially for robot control. NNs are used as classifier and for capturing hand shape in. For features extraction, some methods and algorithms are required even to capture the shape of the hand as applied Gaussian bivariate function for fitting the segmented hand which used to minimise the rotation affection. The selection of specific algorithm for recognition depends on the application needed. In this work application areas for the gestures system are presented. Explanation of gesture recognition issues, detail discussion of recent recognition systems are given as well.

**Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang, and Yu-Bo Yuan (25 june 2014):**

The author in his paper titled Real-Time Hand Gesture Recognition Using Finger Segmentation explained In this work, how the present a novel real-time method for hand gesture recognition. In our framework, the hand region is extracted from the background with the background subtraction method. Then, the palm and fingers are segmented so as to detect and recognize the fingers. Finally, a rule classifier is applied to predict the labels of hand gestures. The experiments on the data set of 1300 images show that our method performs well and is highly efficient. Moreover, our method shows better performance than a state-of-art method on another data set of hand gestures.

**Farah Farhana Mod Ma'asum, Suhana Sulaiman and Azilah Saparon( May,2015 ):**

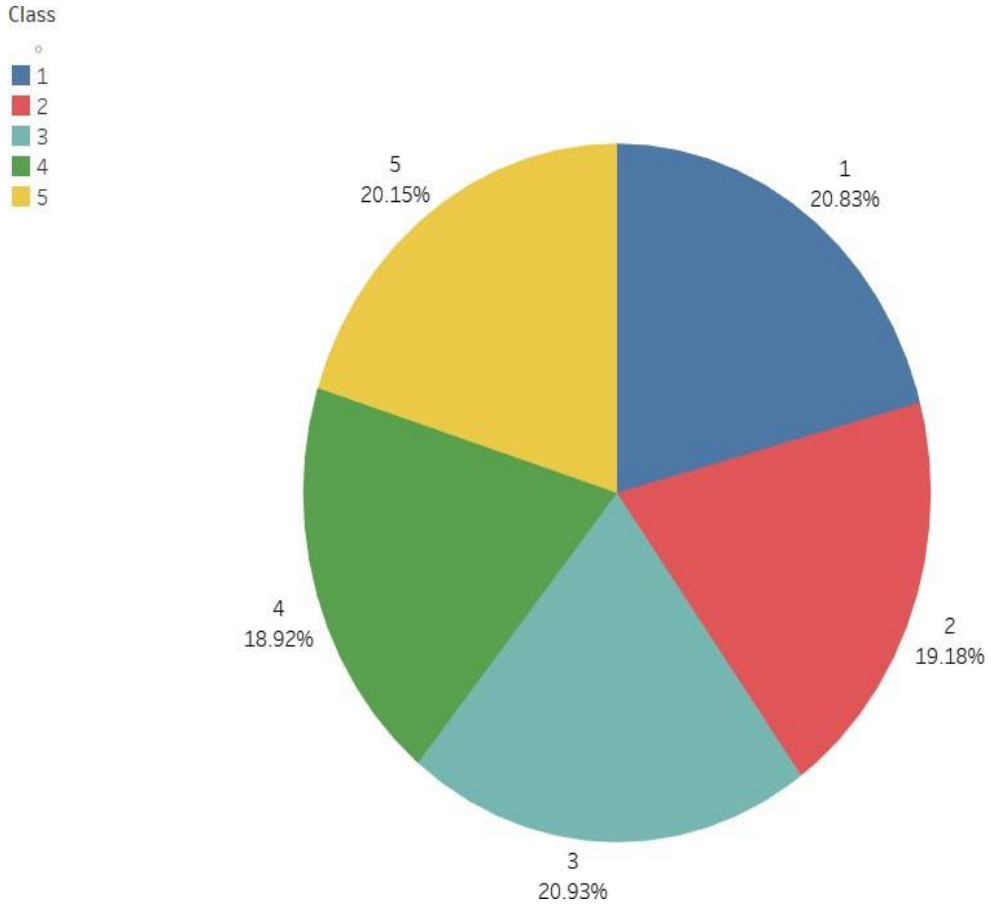
In this paper the author has explained that the hand gesture recognition system has evolved tremendously in the recent few years because of its ability to interact with machine efficiently. Mankind tries to incorporate human gestures into modern technology by searching and finding a replacement of multi touch technology which does not require any touching movement on screen. This paper presents an overview on several methods to realize hand gesture recognition by using three main modules: camera and segmentation module, detection module and feature extraction module. There are many methods which can be used to get the respective results depending on its advantages. Summary of previous research and results of hand gesture methods as well as comparison between gesture recognition are also given in this paper.

**Yunhai Dai, Yanan Xu(2017):**

In the paper titled review of Hand Gesture Recognition Study and Application the author has explained that the Human-computer interaction is an essential part of most people's daily life. The traditional human-computer interaction mode from the original keyboard to the current mouse, joystick, and wireless input devices, greatly facilitates the interaction between people and computers and makes it easier for people to operate the computer and improve work efficiency. However, this kind of interaction mode cannot completely meet the demands of human-computer interaction due to the dependence on additional input hardware devices. Hand gesture can be defined as a variety of gestures or movements produced by hands or arms combined, it is always capable of expressing a signer’s intention, so it can act as a means of natural communication between human and machine. Studies on hand gesture recognition is very important for the development of new human-centered human-computer interaction. This paper reviewed the current study status and application of gesture recognition aiming to summarize the commonly used hand gesture recognition methods, analysis their strength and weak points, and list the challenging problems in current research of hand gesture recognition.

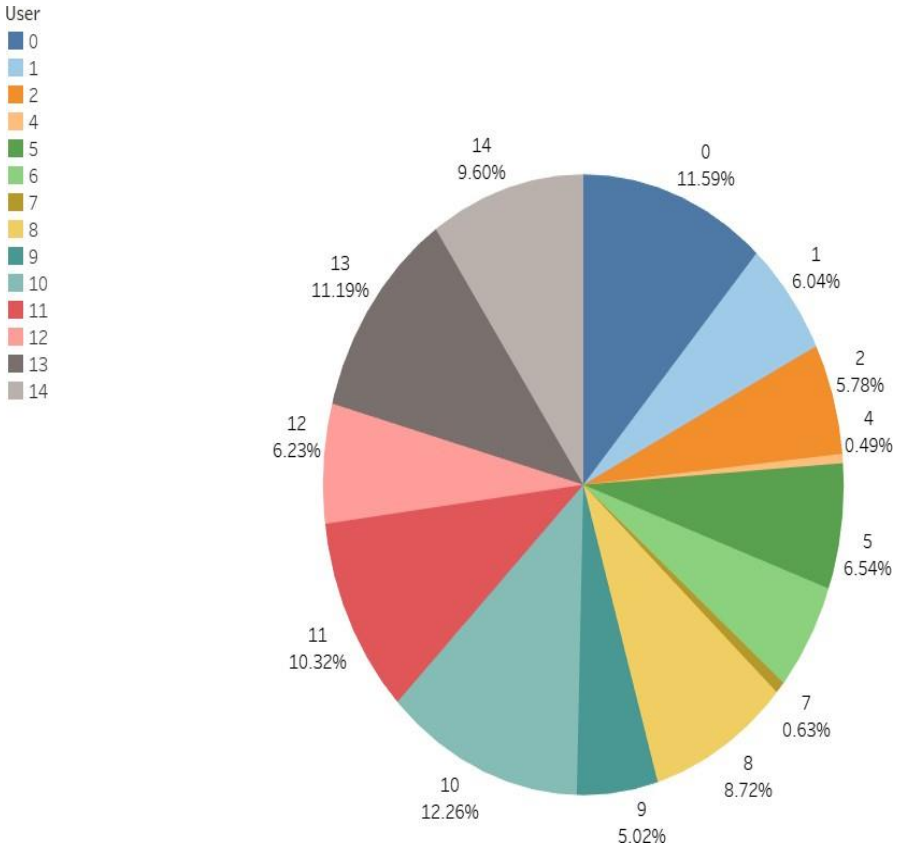
# Data Exploration (EDA)

# Pie Chart



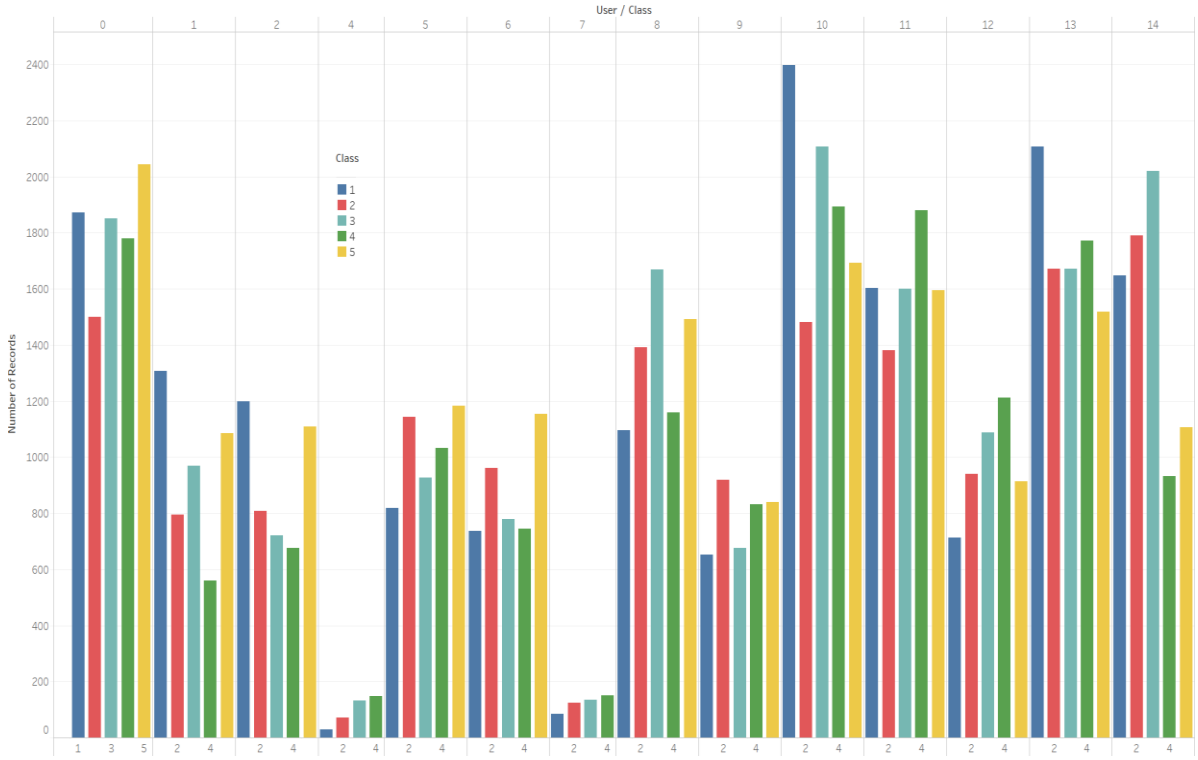

- Here, we used the number of records and percentage of each class (target column) to visualize the usage of each Class using a Pie chart.
- We can also see that all the class are distributed almost equally

# Pie Chart



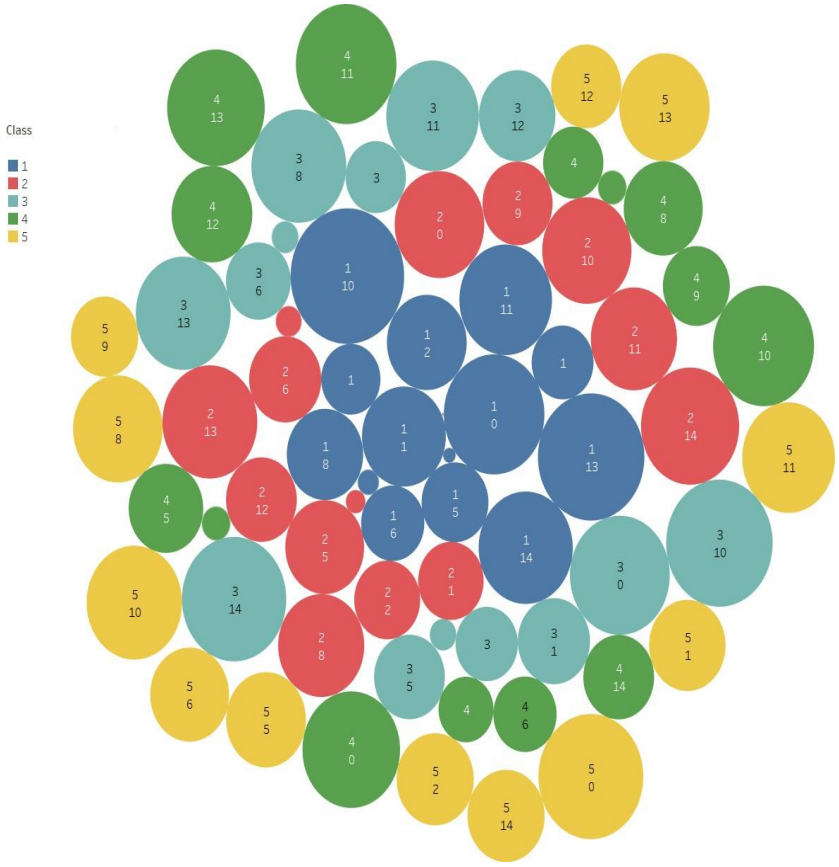

- Here, we took the number of records and the percentage of each user.
- We can see that the user 4 and 7 are having low usage and the user 10 have a higher percentage of 12.26%

# Bar Chart



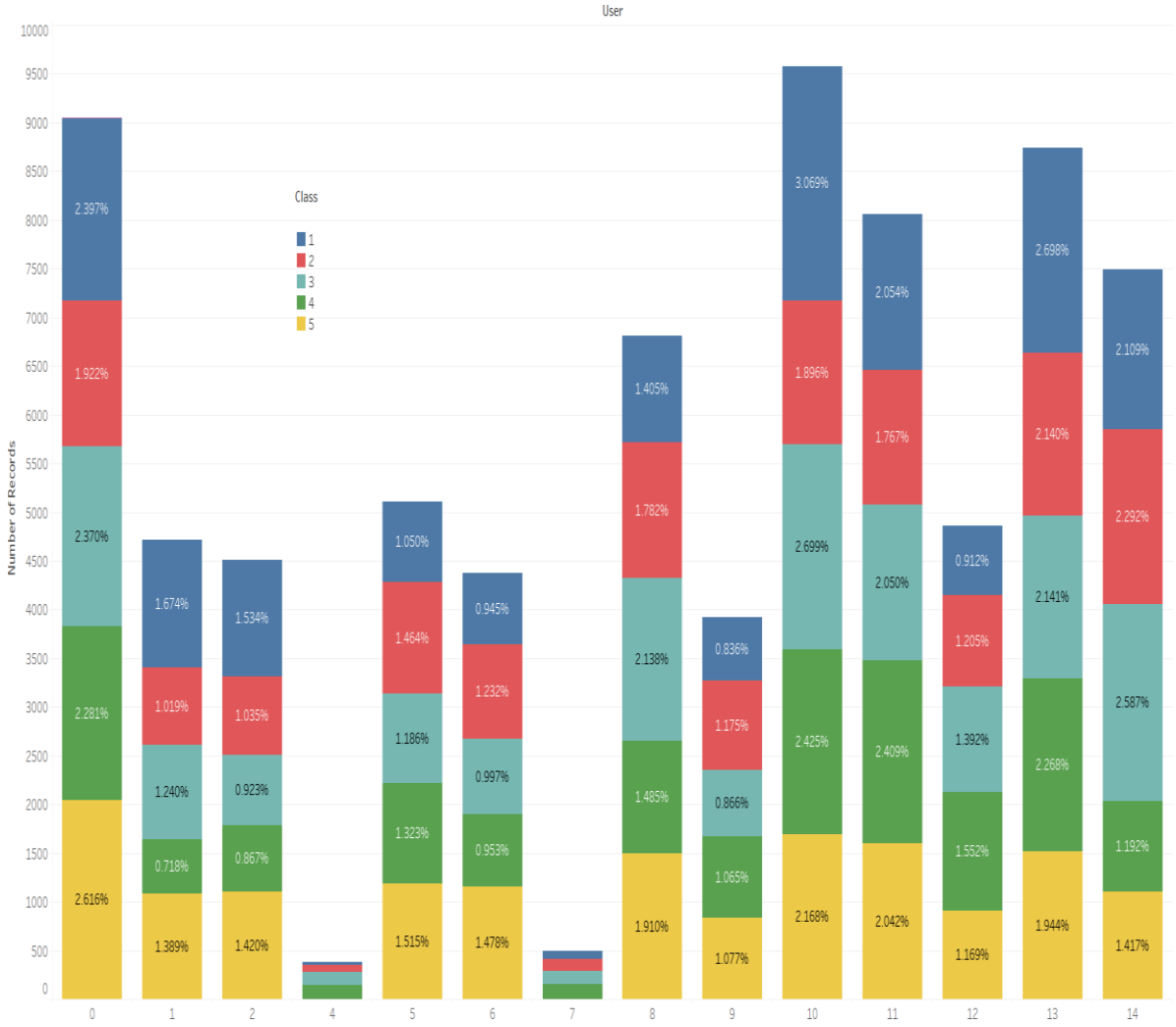

- Here, we took the number of records, user and the class columns to visualize the usage of the each class with respected to the users
- We can see that the user 4 and 7 have the number of records are very low comparing which means the user 4 and 7 are less of the hand gestures.

# Packed Bubble Chart



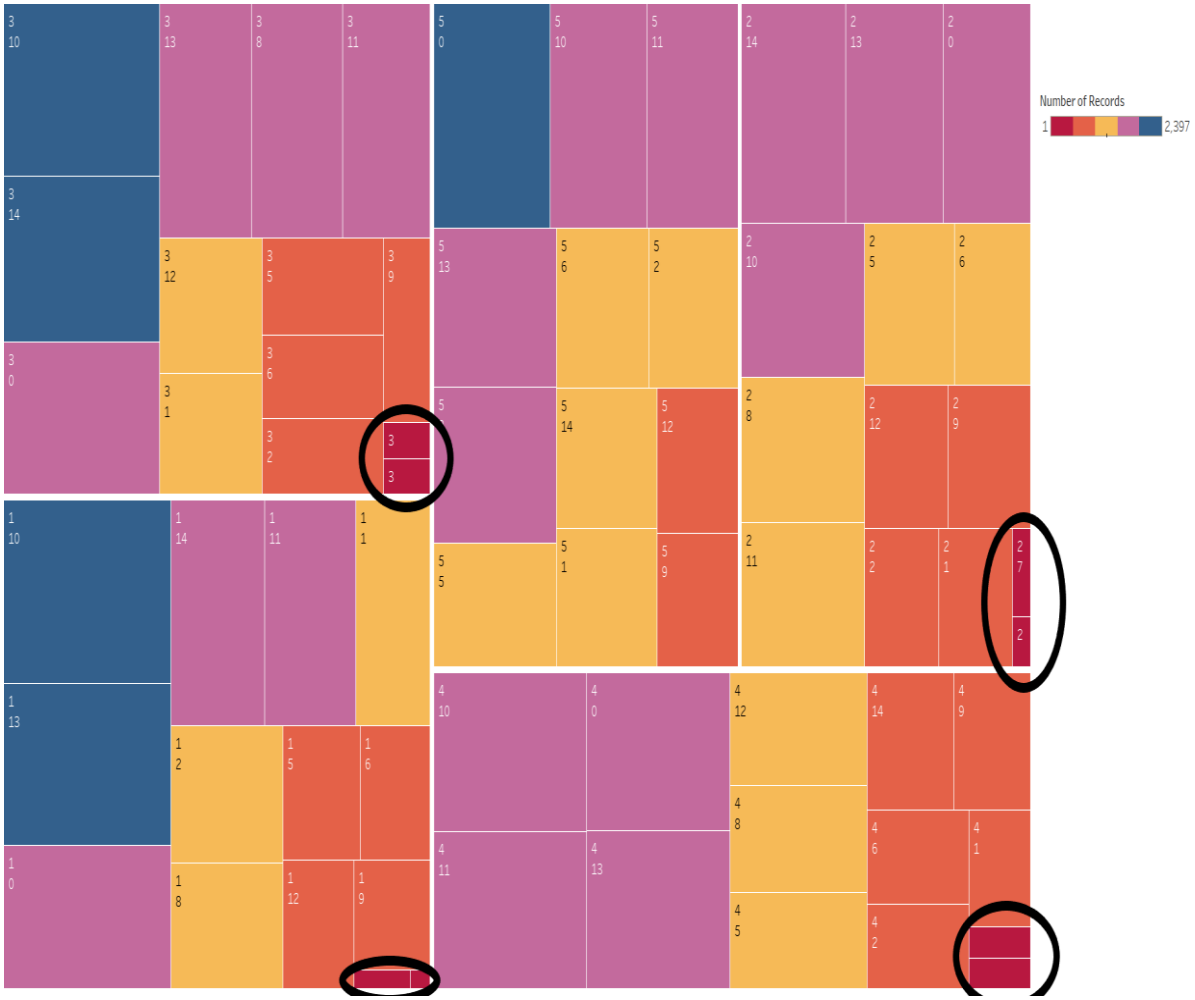

- Here, we took the number of records, user and the class columns to visualize the usage of the each class with respected to the users.
- Here the size represents the number of records and the colour is the class and user is the number
- We can see the small bubbles in the chart which are nothing but a user 4 and 7.

# Stacked Bar Chart



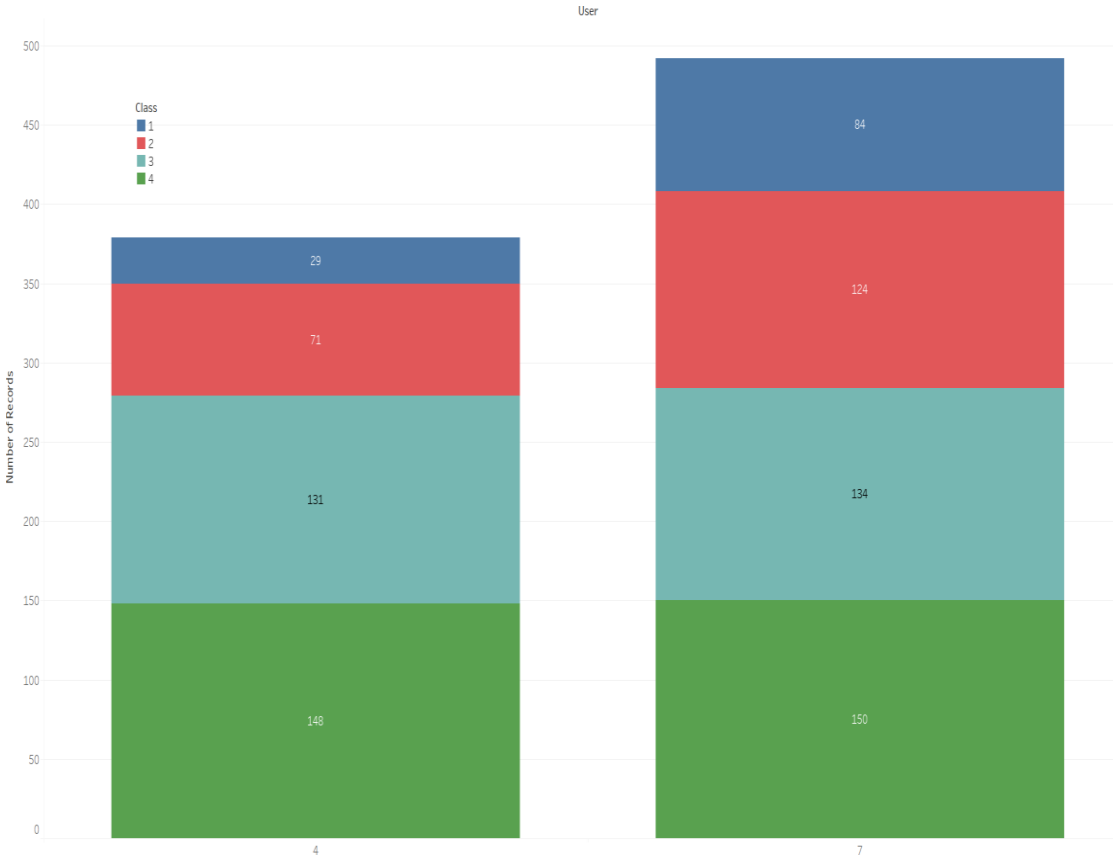

- Here, we took the number of records, user and the class columns to visualize the usage of the each class with respected to the users.
- Here, we can see that the user 10 have the more number of records its above 9500 records.
- And the user 4 and 7 are less then 500 records

# Tree maps




- Here, we took the number of records, user and the class columns to visualize the usage of the each class with respected to the users.
- Here, number of records represents the colour and size which means the blue colour is high number of records and red is less number of records.
- We can see that the black mark in the chart it highlights the red colour which is nothing but the user 4 and 7.

# Stacked Bar Chart




- Here, we took the number of records, user (only 4 and 7) and the class columns to visualize the usage of the each class with respected to the users 4 and 7.
- We can see that user 4 and 7 have not used the class 5(Grab gesture )

# Feature engineering, Model building and Evaluation of model

**ATTEMPTS :**

- Feature Engineering
- Base models

    1. Logistic Regression

    2 .Decision Tree

    3. Random Forest

    4. Naïve Bayes

- Applied PCA.
- Evaluation of Model with and without PCA.
- Parameter Tuning for no._of_Estimators and Learning_rate.
- Performing Ensemble Techniques.

    1.Bagging Classifier

    2.AdaBoost Classifier

    3.RandomForest Classifier

- Clustering (KMeans) (USL).
- Parameter Tuning for optimal number of Clusters.
- Evaluation Of Model with and without PCA after USL.
- Finalizing the Best rightly fit Model.

# Feature Engineering

- To check the significance of each variable, Annova Test has done on few variables.

| | sum_sq | df | F | PR(>F) |
|---|---|---|---|---|
| **Intercept** | 1252.778334 | 1.0 | 702.020009 | 5.243121e-154 |
| **User** | 2.346399 | 1.0 | 1.314853 | 2.515214e-01 |
| **X1** | 383.290207 | 1.0 | 214.784521 | 1.441115e-48 |
| **Y1** | 1290.806499 | 1.0 | 723.329870 | 1.341680e-158 |
| **Z1** | 126.431184 | 1.0 | 70.848305 | 3.921978e-17 |
| **X2** | 0.028912 | 1.0 | 0.016202 | 8.987149e-01 |
| **Y2** | 1624.332715 | 1.0 | 910.228119 | 8.204664e-199 |
| **Z2** | 10.608190 | 1.0 | 5.944516 | 1.476536e-02 |
| **X3** | 0.002700 | 1.0 | 0.001513 | 9.689700e-01 |
| **Y3** | 2732.471735 | 1.0 | 1531.196525 | 0.000000e+00 |
| **Z3** | 294.265454 | 1.0 | 164.897677 | 1.051661e-37 |
| **X4** | 1.671236 | 1.0 | 0.936511 | 3.331797e-01 |
| **Y4** | 6063.415969 | 1.0 | 3397.759376 | 0.000000e+00 |
| **Z4** | 1620.976531 | 1.0 | 908.347413 | 2.080621e-198 |
| **Residual** | 139339.957076 | 78082.0 | NaN | NaN |

**Reason for not dropping columns:**

- As the all variables are co-ordinates or dimensions of a fingers position, if one variable is removed the dimension will change as a result it may effect the other co-ordinates.
- Those indicates the motion of the fingers, one dimension is representing the other.
- One dimension is representing the one movement of the finger. If any disturbance occurred in the dimension, the movement in the finger varies.
- Ideally the performance metrics will decrease and that is the explanation for not dropping the variables which are not significant.

# Model Building and its Explanation

In statistics, the **logistic model** (or **logit model**) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1 and the sum adding to one.

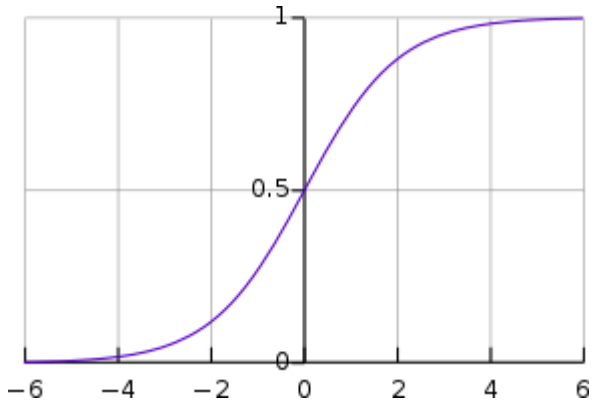
The binary logistic regression model has two levels of the dependent variable: categorical outputs with more than two values are modeled by multinomial logistic regression.

In the multi class logistic regression python Logistic Regression class, multi-class classification can be enabled/disabled by passing values to the argument called ‘‘multi_class’ in the constructor of the algorithm. In the multiclass case, the training algorithm uses the one-vs-rest (OvR) scheme if the ‘multi_class’ option is set to ‘ovr’ and uses the cross-entropy loss if the ‘multi_class’ option is set to ‘multinomial’. (Currently, the ‘multinomial’ option is supported only by the ‘lbfgs’, ‘sag’ and ‘newton-cg’ solvers.) By default, multi_class is set to ’ovr’.

```
In [18]: 1 from sklearn.linear_model import LogisticRegression
         2 lr=LogisticRegression(multi_class='multinomial',solver='lbfgs')
```

**Logistic regression has a sigmoidal curve.**

Following is the graph for the sigmoidal function:




The equation for the sigmoid function is:

$$S(x) = \frac{1}{1 + e^{-x}}$$

It ensures that the generated number is always between 0 and 1 since the numerator is always smaller than the denominator by 1. See below:

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{e^{x} + 1}.$$

The idea in logistic regression is to cast the problem in the form of a generalized linear regression model.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

where ŷ =predicted value, x= independent variables and the β are coefficients to be learned.

This can be compactly expressed in vector form:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$$

Thus, the logistic link function can be used to cast logistic regression into the Generalized Linear Model.

**With respective to Data Set:**

Procedures:

1. Splitting the Data into **X** ('Independent Variables' or 'Predictors') and **y** ('Dependent Variables' or 'Labels')

```
y=df['Class']
X=df.drop('Class',1)
```

2. Performing Train_Test_Split:

   Splitted the Data into Train and Test as X_train, Y_train which contains 70 percent of the data is trained by the model and with the parameter, values which were recorded during the training will be tested on the X_test data which contains 30 percent of the remaining data. That will be recorded as y_prediction.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)
print(X_train.shape)
print(X_test.shape)
print(y_test.shape)
print(y_train.shape)
```

```
(54667, 37)
(23429, 37)
(23429,)
(54667,)
```

3. Fitting all the Supervised Learning Classification Models :

   - Logistic Regression Classifier
   - Decision Tree Classifier
   - Naïve Bayes Classifier
   - Random Forest Classifier ( Ensemble Technique)
   - Bagging Classifier ( It is an Ensemble Technique to enhance the accuracy and to decrease the VARIANCE ERROR, when the model is OVERFITTED)

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import BaggingClassifier
```

- Classifying all model with their Accuracies of Train and Test, Precision, Recall.

```
lr=LogisticRegression()
dt=DecisionTreeClassifier()
rf=RandomForestClassifier()
nb=GaussianNB()
bag=BaggingClassifer()
model=[]
model.append(('Logistic Regression',lr))
model.append(('Decision Tree',dt))
model.append(('Random Forest',rf))
model.append(('Naive Bayes',nb))
model.append(('Bagging Classifier',bag))
cols=[]
train=[]
test=[]
prc=[]
rec=[]
results = []
names = []
bias=[]
variance=[]
scoring = 'f1_weighted'
for k,v in model:
    v.fit(X_train,y_train)
    y_pred=v.predict(X_test)
    Train=v.score(X_train,y_train)
    Test=v.score(X_test,y_test)
    precesion=metrics.precision_score(y_test,y_pred,average='macro')
    recall=precesion=metrics.recall_score(y_test,y_pred,average='macro')
    cols.append(k)
    train.append(Train)
    test.append(Test)
    prc.append(precesion)
    rec.append(recall)
    TEST=pd.DataFrame(test,columns=["TEST_Accuracy"],index=cols)
    TRAIN=pd.DataFrame(train,columns=["TRAIN_Accuracy"],index=cols)
    PRECISION=pd.DataFrame(prc,columns=["PRECISION"],index=cols)
    RECALL=pd.DataFrame(rec,columns=["RECALL"],index=cols)
```

Each and every learning algorithm is declared with some unknown variables like ( ‘lr’,’dt’,’rf’,’nb’,’bag’). All were appended into one variable and Loop was ran. Inside the loop each and every model will fit to the splitting data and will predict the values which are in Multi_class labels. The model will also fit to the Train and Test data to check the accuracies of the splitted data to check whether the model is ‘Overfitting’ or ‘Underfitting’. Later they are going through the performance metrics like ‘Precision’ and ‘Recall’. Whatever the values we obtained , those are all framed as DataFrame and formed a table.

- **Checking for Bias Error and also Variance Error of each model:**

**Justifying Variance and Bias Error**

```
:
results = []
names = []
bias=[]
variance=[]
scoring = 'f1_weighted'
for k,v in model:
    kfold = KFold(n_splits=10, random_state=0,shuffle=True)
    cv_results = cross_val_score(v, X, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    mean=np.mean(results)
    var=np.var(results,ddof=1)
    bias.append(mean)
    variance.append(var)
    names.append(k)
    BIAS=pd.DataFrame(bias,columns=["Bias_Error"],index=names)
    VARIANCE=pd.DataFrame(variance,columns=["Varaince_Error"],index=names)
data1=pd.concat([BIAS,VARIANCE],axis=1,)
#     pd.set_index_name["Model"]
print(data1)
```

Here the data set contains more than 3 samples in the dependent variable which we need to predict, so as per statistics we need to consider the ‘f1_weighted’ as parameter score instead of ‘roc_curve’.

To get the best f1_weighted accuracy , Cross_validation is used and the means of all scores are considered as “Bias Error”, and variance of all scores with degree of freedom equal to 1 is considered as “Variance Error”.

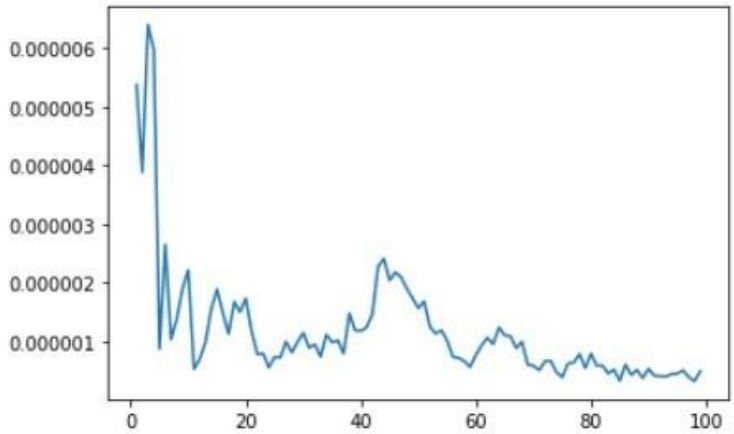
- Tuning Hyper Parameter for the Bagging Classifier to choose the number of estimators by plotting the plot graph or line graph in between the no. of estimators and the variance of Bias Error.

```
bag_var=[]
for val in np.arange(1,100):
    bag=BaggingClassifier(n_estimators=val,random_state=0)
    kfold = KFold(shuffle=True,n_splits=3, random_state=0)
    cv_results = cross_val_score(bag, X, y, cv=kfold, scoring='f1_weighted')
    bag_var.append(np.var(cv_results,ddof=1))
    print(val,np.var(cv_results,ddof=1))
```

```
1 5.371805631712341e-06
2 3.881893691121444e-06
3 6.399020395548554e-06
4 5.9755844625737e-06
5 8.668284391758843e-07
6 2.6478432966428707e-06
7 1.028182777352945e-06
8 1.3748120652196344e-06
9 1.896577845981786e-06
10 2.2128147319821046e-06
11 5.236391690922059e-07
12 6.98232853671389e-07
13 9.836543315438562e-07
14 1.5475424318393354e-06
15 1.885591975568534e-06
16 1.4846306203430361e-06
17 1.122240593722175e-06
93 3.949177583035689e-07
94 4.368794254420576e-07
95 4.455396954484868e-07
96 4.994717605235174e-07
97 3.831042587733308e-07
98 3.164541803916613e-07
99 4.879222266783262e-07
```

```
x_axis=np.arange(1,100)
plt.plot(x_axis,bag_var)
```

```
[<matplotlib.lines.Line2D at 0x264d87cf128>]
```




```
bag=BaggingClassifier(n_estimators=85,random_state=0)
```

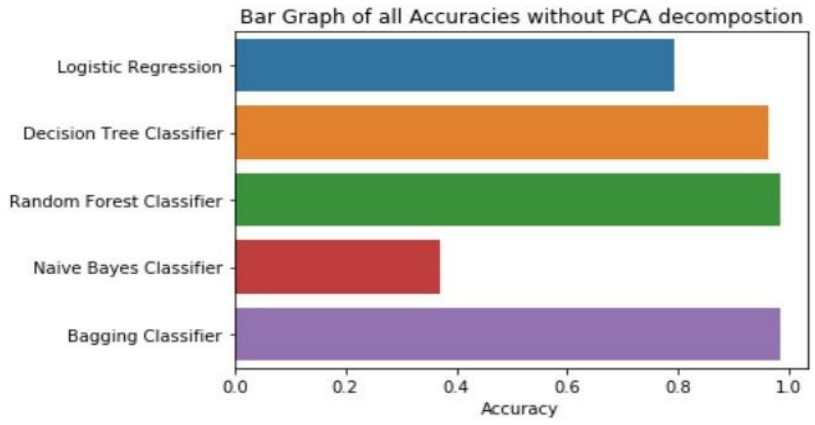
We can see in the graph that, at 85th (approx.) estimator there is an reduce in the variance in the model and decreased gradually. So, considering the n_estimators as 85 and builded a model to figure out the Bias Error and Variance Error.

| | Bias_Error | Varaince_Error |
|---|---|---|
| Logistic Regression | 0.777002 | 0.000022 |
| Decision Tree | 0.873634 | 0.009842 |
| Random Forest | 0.910567 | 0.009270 |
| Naive Bayes | 0.746509 | 0.089722 |
| Bagging_Classifier | 0.794560 | 0.080835 |

Accuracies of each Model:

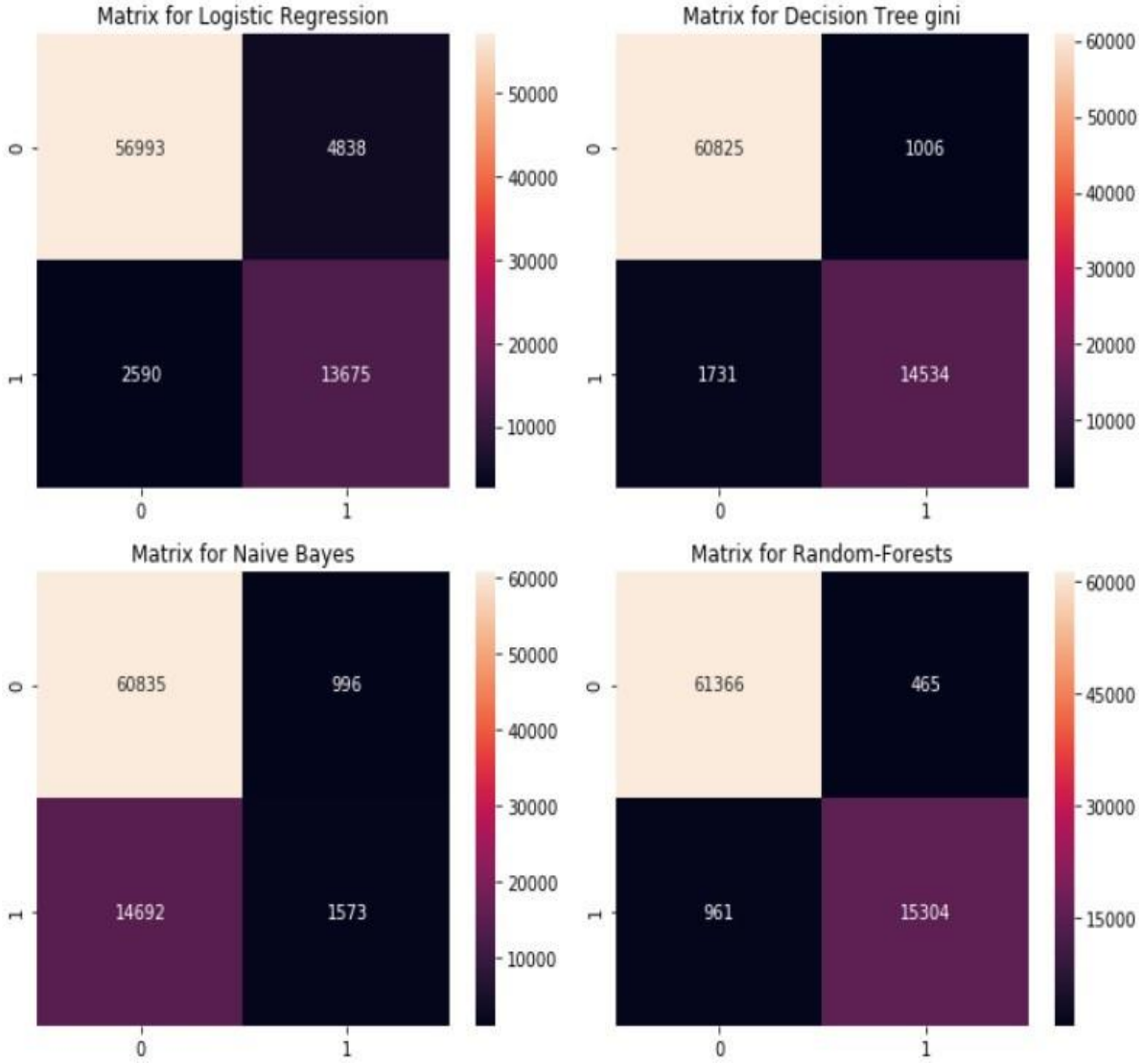
| Without Decompostion | | | | | | |
|---|---|---|---|---|---|---|
| | TRAIN_Accuracy | TEST_Accuracy | PRECISION | RECALL | Bias_Error | Varaince_Error |
| **Logistic Regression** | 0.777909891 | 0.775748005 | 0.77517771 | 0.77517771 | 0.777001755 | 2.23E-05 |
| **Decision Tree** | 1 | 0.963336037 | 0.962683048 | 0.962683048 | 0.873634417 | 0.009841557 |
| **Random Forest** | 0.999634149 | 0.983268599 | 0.982937569 | 0.982937569 | 0.91056673 | 0.00927049 |
| **Naive Bayes** | 0.372363583 | 0.370054206 | 0.36216218 | 0.36216218 | 0.746509384 | 0.089721559 |
| **Bagging_Classifier** | 1 | 0.984847838 | 0.984557818 | 0.984557818 | 0.794559897 | 0.080835263 |

Graphical Representation of each model:



```
Logistic Regression : 0.7937171881002176
Decision Tree Classification : 0.9646591830637244
Random Forest Classification : 0.9834393273293781
Naive Bayes Classifier: 0.3700542063254941
Bagging Classifier: 0.9842502881044859
```

4. Confusion Matrix of all the Algorithms:

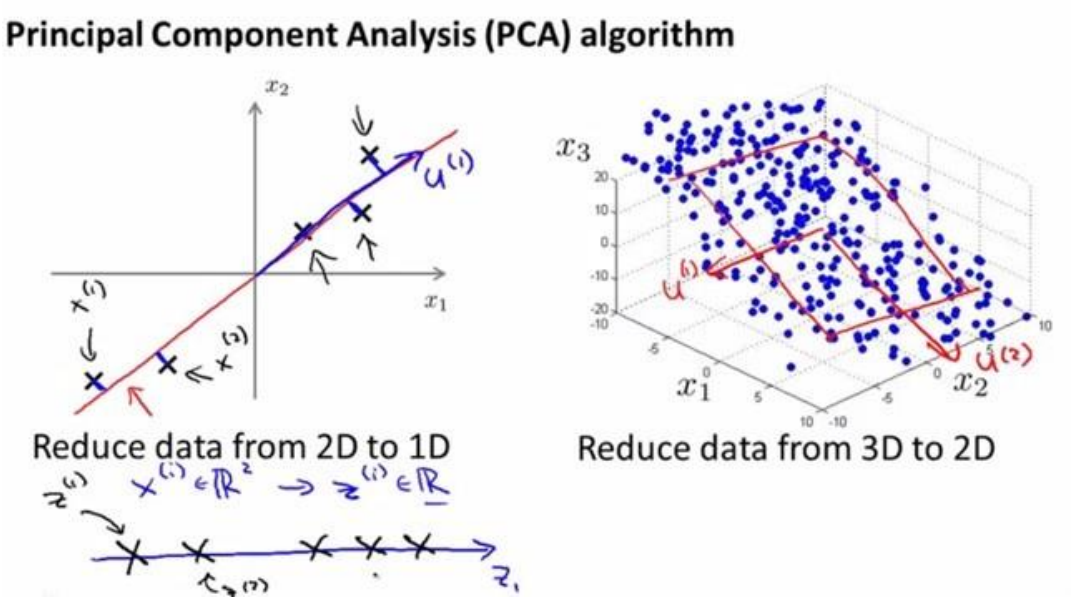


**PCA (PRINCIPLE COMPONENT ANALYSIS):**

- The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other, either heavily or lightly, while retaining the variation present in the dataset, up to the maximum extent. The same is done by transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order. So, in this way, the 1st principal component retains maximum variation that was present in the original components. The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.

- Importantly, the dataset on which PCA technique is to be used must be scaled. The results are also sensitive to the relative scaling. As a layman, it is a method of summarizing data. Imagine some wine bottles on a dining table. Each wine is described by its attributes like colour, strength, age, etc. But redundancy will arise because many of them will measure related properties. So what PCA will do in this case is summarize each wine in the stock with less characteristics.

- Intuitively, Principal Component Analysis can supply the user with a lower-dimensional picture, a projection or "shadow" of this object when viewed from its most informative viewpoint.



- The data which we have the columns are all with continuous values as they are co-ordinates of 12 markers of hand posture of five fingers. So, our data doesn't need to scale when applying PCA side by the existence of multi collinearity is less, we are applying PCA to reduce the columns and the complexity of the model.

- **Properties of Principal Component**

Technically, a principal component can be defined as a linear combination of optimally-weighted observed variables. The output of PCA are these principal components, the number of which is less than or equal to the number of original variables. Less, in case when we wish to discard or reduce the dimensions in our dataset. The PCs possess some useful properties which are listed below:

1. The PCs are essentially the linear combinations of the original variables, the weights vector in this combination is actually the eigenvector found which in turn satisfies the principle of least squares.

2. The PCs are orthogonal, as already discussed.

3. The variation present in the PCs decrease as we move from the 1st PC to the last one, hence the importance.

The least important PCs are also sometimes useful in regression, outlier detection, etc.

**Implementing PCA on a Mutli-Dimension Dataset:**

**For instance, let us go with Implementing PCA on 2D-Dataset:**

**Step 1: Normalize the data:**

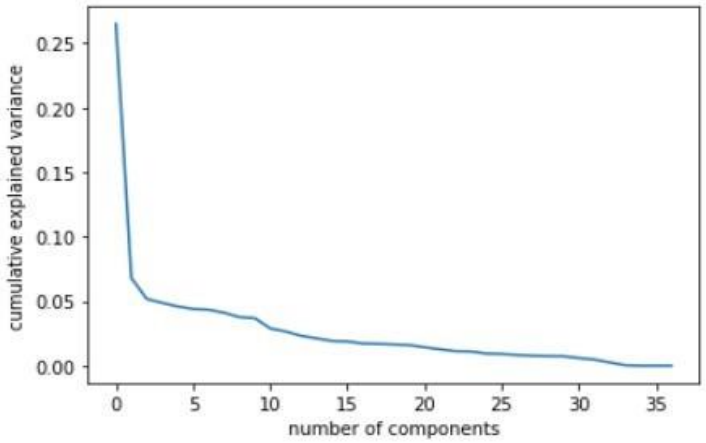
First step is to normalize the data that we have so that PCA works properly. This is done by subtracting the respective means from the numbers in the respective column. So if we have two dimensions X and Y, all X become X- and all Y become y-. This produces a dataset whose mean is zero.

```
1  from sklearn.decomposition import PCA
```

```
1  pca=PCA()
2  X_train=pca.fit_transform(X_train)
3  X_test=pca.transform(X_test)
```

2. Judging the number of principal components based on the contrast between the 'Explained Variance Ratio' and "Number of Principal Components"

In [26]:
```
#Explained variance
plt.plot(pca.explained_variance_ratio_)
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
plt.savefig("Principle Components")
```




```
<Figure size 432x288 with 0 Axes>
```

In [27]:
```
pca.explained_variance_ratio_
```

Out[27]:
```
array([2.65070928e-01, 6.80180154e-02, 5.17449822e-02, 4.87626504e-02,
       4.59751684e-02, 4.39881008e-02, 4.33926290e-02, 4.10994575e-02,
       3.77359225e-02, 3.69274220e-02, 2.89133752e-02, 2.65928964e-02,
       2.31912080e-02, 2.12057969e-02, 1.91775780e-02, 1.88299864e-02,
       1.72720355e-02, 1.69753870e-02, 1.65152795e-02, 1.60271449e-02,
       1.43266159e-02, 1.27819748e-02, 1.12629447e-02, 1.10342636e-02,
       9.43729194e-03, 9.09174563e-03, 8.32447160e-03, 7.73790072e-03,
       7.44279407e-03, 7.36219814e-03, 5.94996700e-03, 4.81373805e-03,
       2.56752690e-03, 4.08866367e-04, 3.42237814e-05, 7.30335214e-06,
       2.09498641e-07])
```

**Step 2: Calculate the covariance matrix :**

Since the dataset we considered is 2-dimensional, this will result in a 2x2 Covariance matrix.

$$Matrix(Covariance) = \begin{bmatrix} Var[X_1] & Cov[X_1, X_2] \\ Cov[X_2, X_1] & Var[X_2] \end{bmatrix}$$

Please note that $Var[X_1] = Cov[X_1,X_1]$ and $Var[X_2] = Cov[X_2,X_2]$.

```
1
2 # covariance matrix
3 import numpy as np
4 cov_matrix = np.cov(X_train.T)
5 print('Covariance Matrix \n%s', cov_matrix)
```

```
Covariance Matrix
%s [[ 1.21033288e+04 -2.62150626e-12 -3.44991076e-13 ...  5.32393635e-16
  -2.07966264e-16 -8.31865055e-17]
 [-2.62150626e-12  3.10575140e+03 -2.29994050e-13 ... -8.31865055e-16
  -5.82305538e-17  4.78322407e-17]
 [-3.44991076e-13 -2.29994050e-13  2.36271303e+03 ... -9.98238066e-17
   2.24603565e-16 -2.49559516e-17]
 ...
 [ 5.32393635e-16 -8.31865055e-16 -9.98238066e-17 ...  1.56268242e+00
   1.16461108e-16 -1.45576385e-17]
 [-2.07966264e-16 -5.82305538e-17  2.24603565e-16 ...  1.16461108e-16
   3.33476300e-01  2.91152769e-17]
 [-8.31865055e-17  4.78322407e-17 -2.49559516e-17 ... -1.45576385e-17
   2.91152769e-17  9.56585830e-03]]
```

**Step 3: Calculate the eigenvalues and eigenvectors:**

Next step is to calculate the eigenvalues and eigenvectors for the covariance matrix. The same is possible because it is a square matrix. $\lambda$ is an eigenvalue for a matrix $A$ if it is a solution of the characteristic equation:

$$det( \lambda I - A ) = 0$$

Where, $I$ is the identity matrix of the same dimension as $A$ which is a required condition for the matrix subtraction as well in this case and '*det*' is the determinant of the matrix. For each eigenvalue $\lambda$, a corresponding eigen-vector $v$, can be found by solving:

$$( \lambda I - A )v = 0$$

```
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
```

```
eig_vals
```

```
array([1.21033288e+04, 3.10575140e+03, 2.36271303e+03, 2.22653762e+03,
       2.09925919e+03, 2.00852826e+03, 1.98133859e+03, 1.87663073e+03,
       1.68613259e+03, 1.72304931e+03, 1.32020546e+03, 1.21425074e+03,
       1.05892720e+03, 9.68271908e+02, 1.86691318e+01, 1.56268242e+00,
       3.33476300e-01, 9.56585830e-03, 1.17235121e+02, 8.75661977e+02,
       8.59790694e+02, 7.88653537e+02, 7.75108355e+02, 7.31810939e+02,
       7.54099517e+02, 6.54163563e+02, 2.19798734e+02, 5.83634144e+02,
       2.71679764e+02, 5.14274137e+02, 5.03832397e+02, 4.30913524e+02,
       4.15135631e+02, 3.80101348e+02, 3.53318100e+02, 3.39843319e+02,
       3.36163251e+02])
```

```
eig_vecs
```

```
array([[ 1.00000000e+00, -2.91356900e-16, -3.54177891e-17, ...,
         1.37742568e-17,  6.51717414e-18, -1.60616199e-18],
       [ 0.00000000e+00, -1.00000000e+00, -6.21098152e-16, ...,
         3.38471635e-18,  7.57579478e-18,  2.46687863e-17],
       [ 0.00000000e+00, -1.92636818e-16, -1.00000000e+00, ...,
         5.36420947e-17,  6.99943966e-18,  3.96552780e-17],
       ...,
       [ 0.00000000e+00,  2.20841512e-19,  1.32667662e-18, ...,
        -2.28638704e-16, -7.94588047e-17,  2.21015844e-17],
       [ 0.00000000e+00,  3.51029229e-20, -9.67432851e-19, ...,
        -5.52414787e-17, -2.18379159e-16,  2.34391902e-17],
       [ 0.00000000e+00,  5.62051565e-21, -1.80291513e-19, ...,
        -5.84535128e-17,  4.00158855e-17,  7.98487019e-17]])
```

**Step 4: Choosing components and forming a feature vector:**

We order the eigenvalues from largest to smallest so that it gives us the components in order or significance. Here comes the dimensionality reduction part. If we have a dataset with *n* variables, then we have the corresponding *n* eigenvalues and eigenvectors. It turns out that the eigenvector corresponding to the highest eigenvalue is the principal component of the dataset and it is our call as to how many eigenvalues we choose to proceed our analysis with. To reduce the dimensions, we choose the first *p* eigenvalues and ignore the rest. We do lose out some information in the process, but if the eigenvalues are small, we do not lose much.

Next we form a feature vector which is a matrix of vectors, in our case, the eigenvectors. In fact, only those eigenvectors which we want to proceed with. Since we just have 2 dimensions in the running example, we can either choose the one corresponding to the greater eigenvalue or simply take both.

*Feature Vector = ($eig_1$, $eig_2$)*

```
tot = sum(eig_vals)
var_exp = [( i /tot ) * 100 for i in sorted(eig_vals, reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained", cum_var_exp)
```

```
Cumulative Variance Explained [ 26.50709279  33.30889432  38.48339255  43.35965758  47.95717442
  52.3559845   56.69524741  60.80519316  64.57878541  68.27152761
  71.16286513  73.82215477  76.14127557  78.26185526  80.17961306
  82.06261171  83.78981525  85.48735396  87.13888191  88.7415964
  90.17425799  91.45245547  92.57874993  93.68217629  94.62590549
  95.53508005  96.36752721  97.14131728  97.88559669  98.62181651
  99.21681321  99.69818701  99.9549397   99.99582634  99.99924871
  99.99997905 100.        ]
```

```

len(cum_var_exp[cum_var_exp>99])
```

```
7
```

```
len(cum_var_exp)
```

```
37
```

As per above cumulative variance Explained values, there are 37 columns excluding the target variable, in that there are 7 columns which are not significantly varied, dropping out those and with including the 30 columns as PRINCPLE COMPONENTS we are fitting the model.

**Step 5: Forming Principal Components:**

This is the final step where we actually form the principal components using all the math we did till here. For the same, we take the transpose of the feature vector and left-multiply it with the transpose of scaled version of original dataset.

$$NewData = FeatureVector^T \times ScaledData^T$$

Here,

*NewData* is the Matrix consisting of the principal components,

*FeatureVector* is the matrix we formed using the eigenvectors we chose to keep, and

*ScaledData* is the scaled version of original dataset

```
pca1 =PCA(n_components=30)
X_train1=pca1.fit_transform(X_train)
X_test1=pca1.transform(X_test)
```

```
pca1
```

```
PCA(copy=True, iterated_power='auto', n_components=30, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

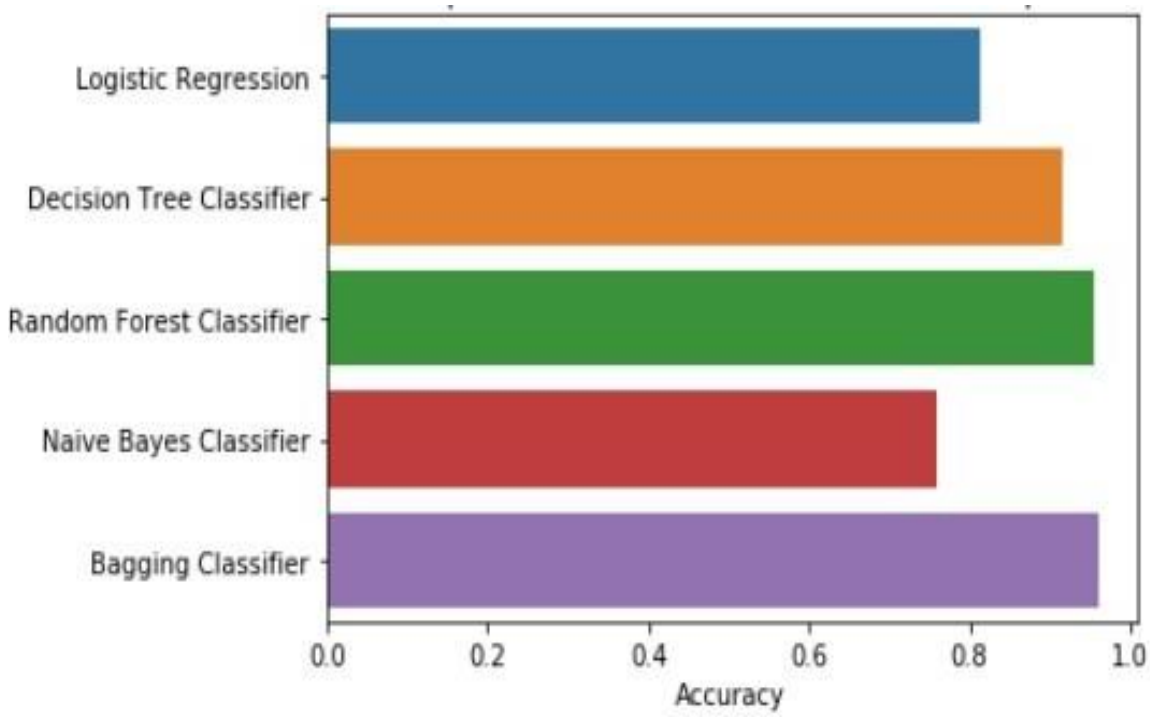
5. Fitting all the Supervised Learning Classification Models :

    - Logistic Regression Classifier
    - Decision Tree Classifier
    - Naïve Bayes Classifier
    - Random Forest Classifier ( Ensemble Technique)
    - Bagging Classifier ( It is an Ensemble Technique to enhance the accuracy and to decrease the VARIANCE ERROR, when the model is OVERFITTED)

| Before Clustering With PCA | | | | | | |
|---|---|---|---|---|---|---|
| | TRAIN_Accuracy | TEST_Accuracy | PRECISION | RECALL | Bias_Error | Varaince_Error |
| Logistic Regression | 0.838055866 | 0.839344402 | 0.839054179 | 0.839054179 | 0.777001755 | 2.23E-05 |
| Decision Tree | 1 | 0.912672329 | 0.759716945 | 0.759716945 | 0.873634417 | 0.009841557 |
| Random Forest | 0.999268297 | 0.957104443 | 0.956519883 | 0.956519883 | 0.910651745 | 0.009283446 |
| Naive Bayes | 0.759068542 | 0.7585471 | 0.759858348 | 0.759858348 | 0.746573145 | 0.089752652 |
| Bagging_Classifier | 1 | 0.960390968 | 0.959778664 | 0.959778664 | 0.794610906 | 0.080855009 |

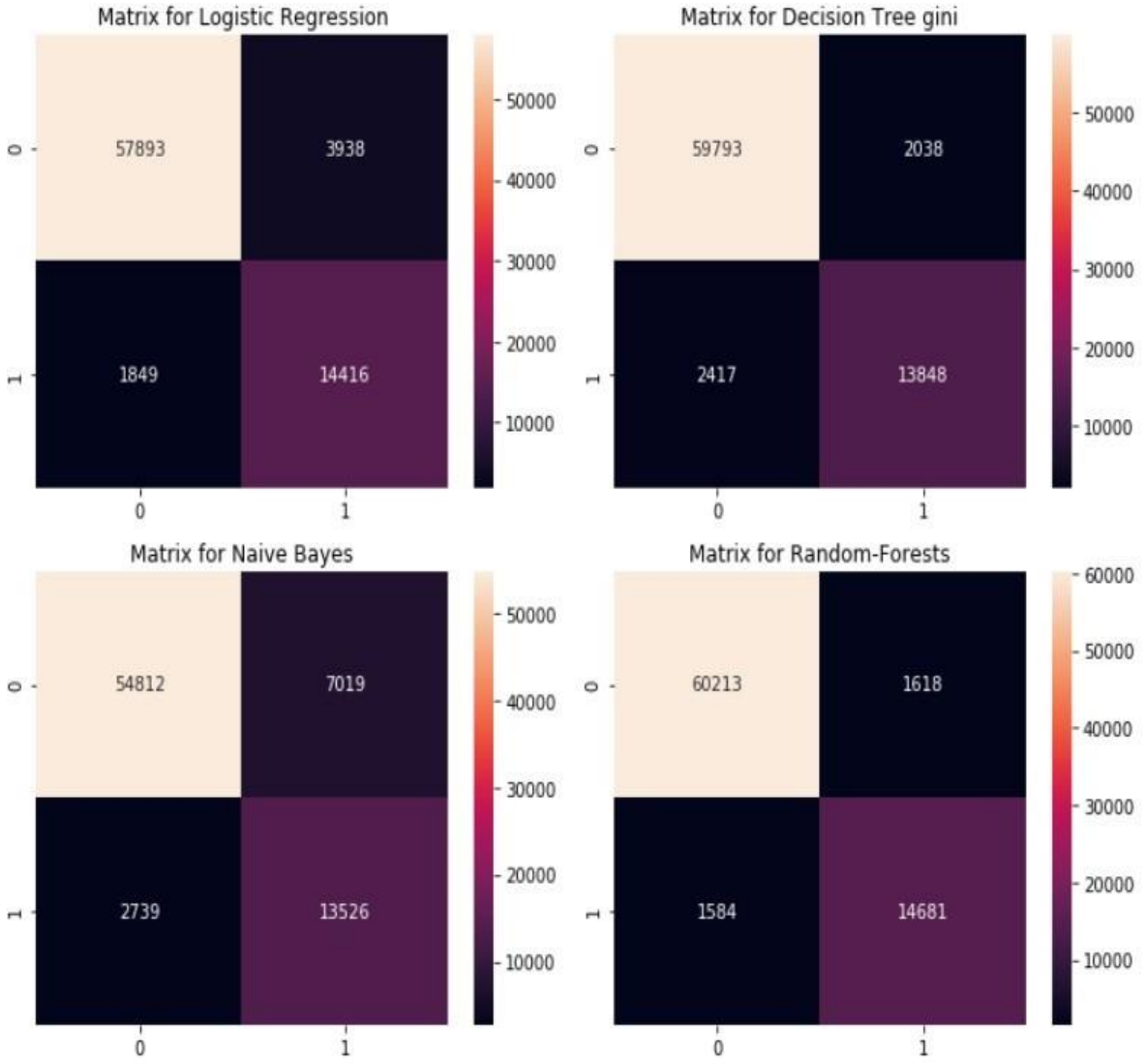
As here we can see the accuracy increases after applying PCA, as the no. of principal components are 30. It doesn't mean that we are leaving the other 7 columns. We are compressing the 37 columns of 37 dimensional data into 30 dimensional data, so that the complexity of the model will some what decreases and the variance error will be decrease compare to before PCA and also the performance parameter's accuracies like RECALL and PRECSION are also increased drastically. Thus applying the PCA with Logistic Regression shows the better accuracy without highly biased, not having high variance error , neither highly overfitted not underfitted, since it is a rightly fitted and having a high precision accuracy value, we prefer the LOGISTIC REGRESSION AFTER APPLYING PCA for this DataSet.

**The graphical representation of all model's accuracies:**




```
Logistic Regression : 0.8116010072986469
Decision Tree Classification : 0.9144649792991592
Random Forest Classification : 0.9549703359084895
Naive Bayes Classifier: 0.7585470997481754
Bagging Classifier: 0.9600495112894276
```

- **Confusion Matrix after applying PCA:**




- **Comparison of accuracies of all models with and without PCA:**

| Before Clustering With No PCA | | | | | | |
|---|---|---|---|---|---|---|
| | TRAIN_Accuracy | TEST_Accuracy | PRECISION | RECALL | Bias_Error | Varaince_Error |
| Logistic Regression | 0.777909891 | 0.775748005 | 0.77517771 | 0.77517771 | 0.777001755 | 2.23E-05 |
| Decision Tree | 1 | 0.963336037 | 0.962683048 | 0.962683048 | 0.873634417 | 0.009841557 |
| Random Forest | 0.999634149 | 0.983268599 | 0.982937569 | 0.982937569 | 0.91056673 | 0.00927049 |
| Naive Bayes | 0.372363583 | 0.370054206 | 0.36216218 | 0.36216218 | 0.746509384 | 0.089721559 |
| Bagging_Classifier | 1 | 0.984847838 | 0.984557818 | 0.984557818 | 0.794559897 | 0.080835263 |

| Before Clustering With PCA | | | | | | |
|---|---|---|---|---|---|---|
| | TRAIN_Accuracy | TEST_Accuracy | PRECISION | RECALL | Bias_Error | Varaince_Error |
| Logistic Regression | 0.838055866 | 0.839344402 | 0.839054179 | 0.839054179 | 0.777001755 | 2.23E-05 |
| Decision Tree | 1 | 0.912672329 | 0.759716945 | 0.759716945 | 0.873634417 | 0.009841557 |
| Random Forest | 0.999268297 | 0.957104443 | 0.956519883 | 0.956519883 | 0.910651745 | 0.009283446 |
| Naive Bayes | 0.759068542 | 0.7585471 | 0.759858348 | 0.759858348 | 0.746573145 | 0.089752652 |
| Bagging_Classifier | 1 | 0.960390968 | 0.959778664 | 0.959778664 | 0.794610906 | 0.080855009 |

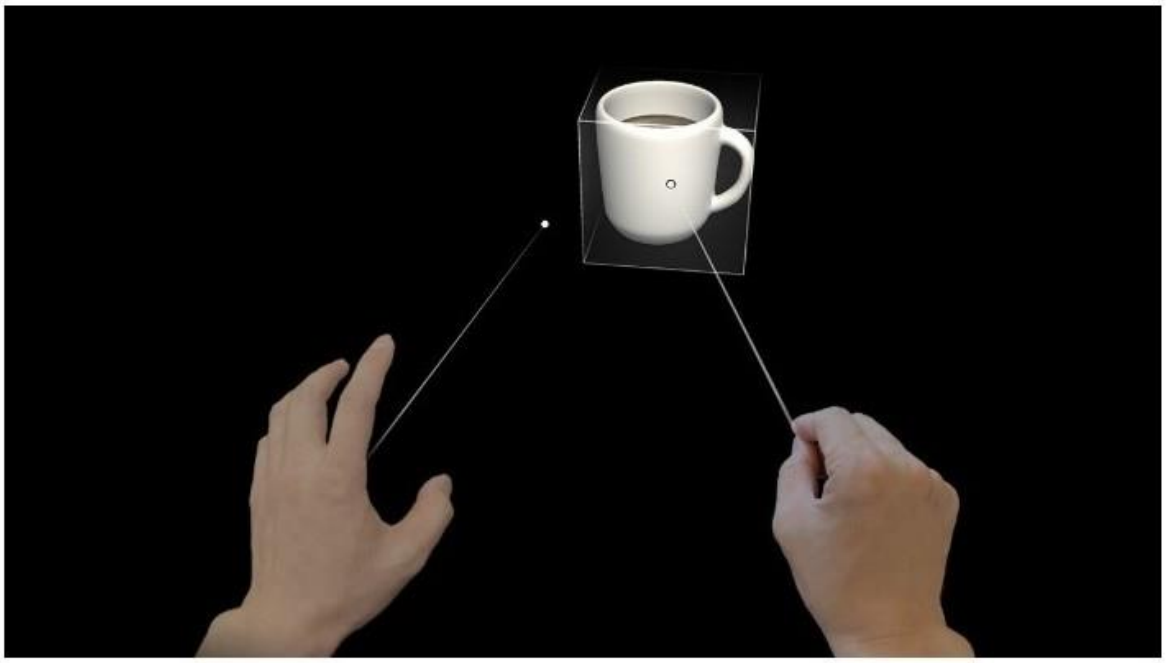
As here we can see the accuracy increases after applying PCA, as the no. of principal components are 30.

Thus applying the PCA with Logistic Regression shows the better accuracy without highly biased, not having high variance error , neither highly overfitted not underfitted, since it is a rightly fitted and having a high precision accuracy value, we prefer the LOGISTIC REGRESSION AFTER APPLYING PCA for this Data Set.

# Business Recommendations & Future enhancements

**Holograms (Mixed Reality platform):**

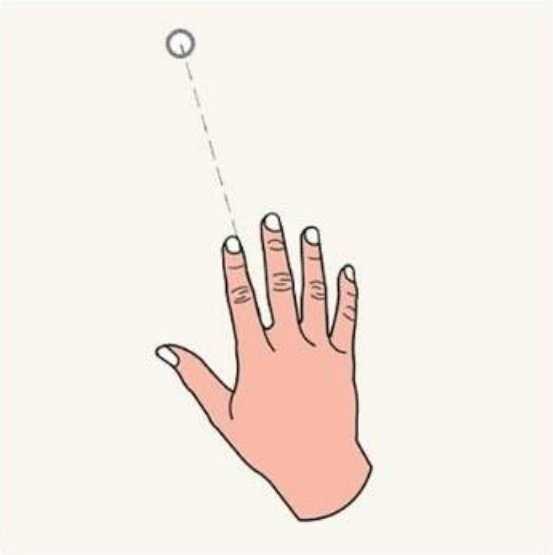
**Point and commit with hands**

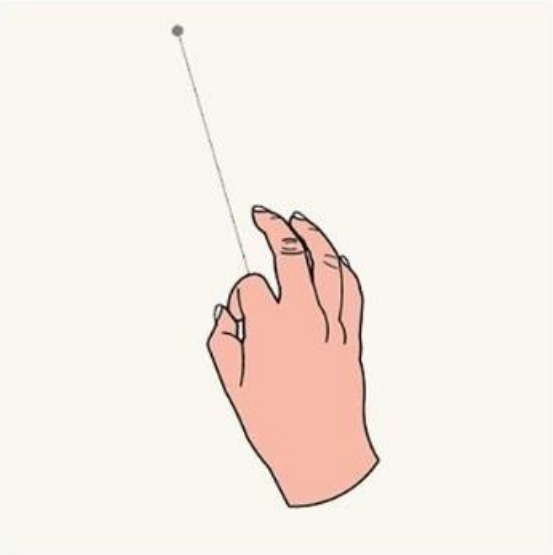


Point and commit with hands is an input model that enables users to target, select and manipulate 2D content and 3D objects that are out of reach. This "far" interaction technique is unique to mixed reality, and is not a way humans naturally interact with the real world. For example, in the super hero movie, X-Men, the character Magneto is capable of reaching out and manipulating a far object in the distance with his hands. This is not something humans can do in reality. In both HoloLens (AR) and Mixed Reality (MR), we equip users with this magical power, breaking the physical constraint of the real world, not only to have a fun experience with holographic content but also to make user interactions more effective and efficient

Hand rays

On HoloLens 2, we created a hand ray that shoots out from the center of the user's palm. This ray is treated as an extension of the hand. A donut-shaped cursor is attached to the end of the ray to indicate the location where the ray intersects with a target object. The object that the cursor lands on can then receive gestural commands from the hand.





Business uses :

- Machinery study and manufacturing ( which increse the productions ).
- Increase speed of work.
- Training uses.

**Automobile sector**

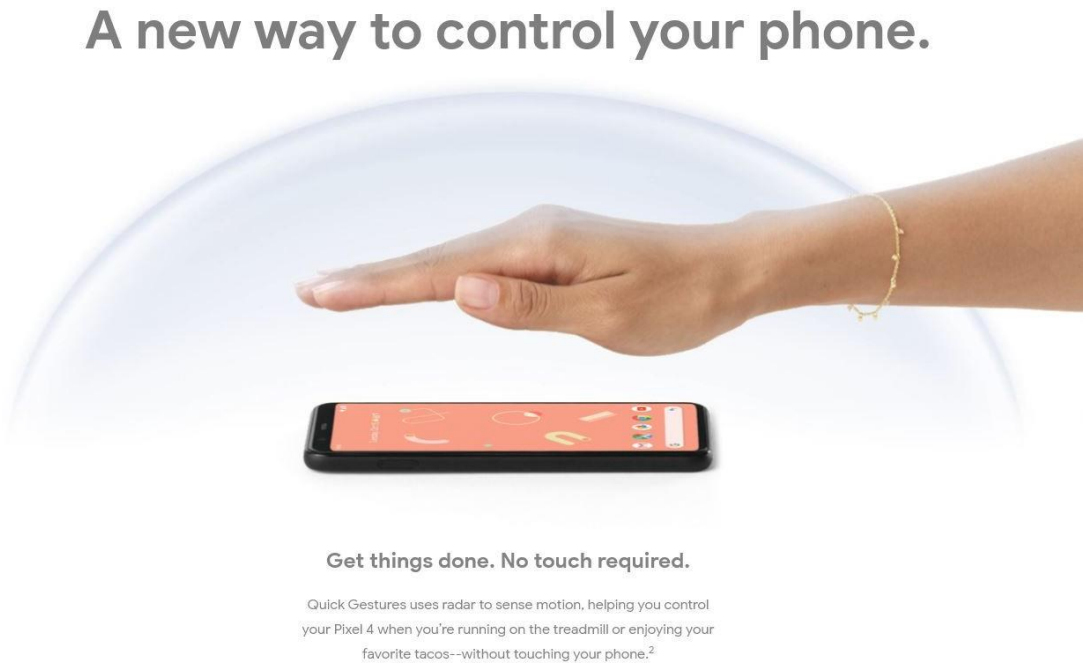
Controlling multiple devices while driving steals drivers' attention from the road and is becoming the cause of accidents in 1 out of 3 cases. Many research efforts are being dedicated to design, manufacture and test Human-Machine Interfaces that allow operating car devices without distracting the drivers' attention. A complete system for controlling the infotainment equipment through hand gestures is explained in this paper. The system works with a visible-infrared camera mounted on the ceiling of the car and pointing to the shift-stick area, and is based in a combination of some new and some well-known computer vision algorithms. The system has been tested by 23 volunteers on a car simulator and a real vehicle and the results show that the users slightly prefer this system to an equivalent one based on a touch-screen interface.

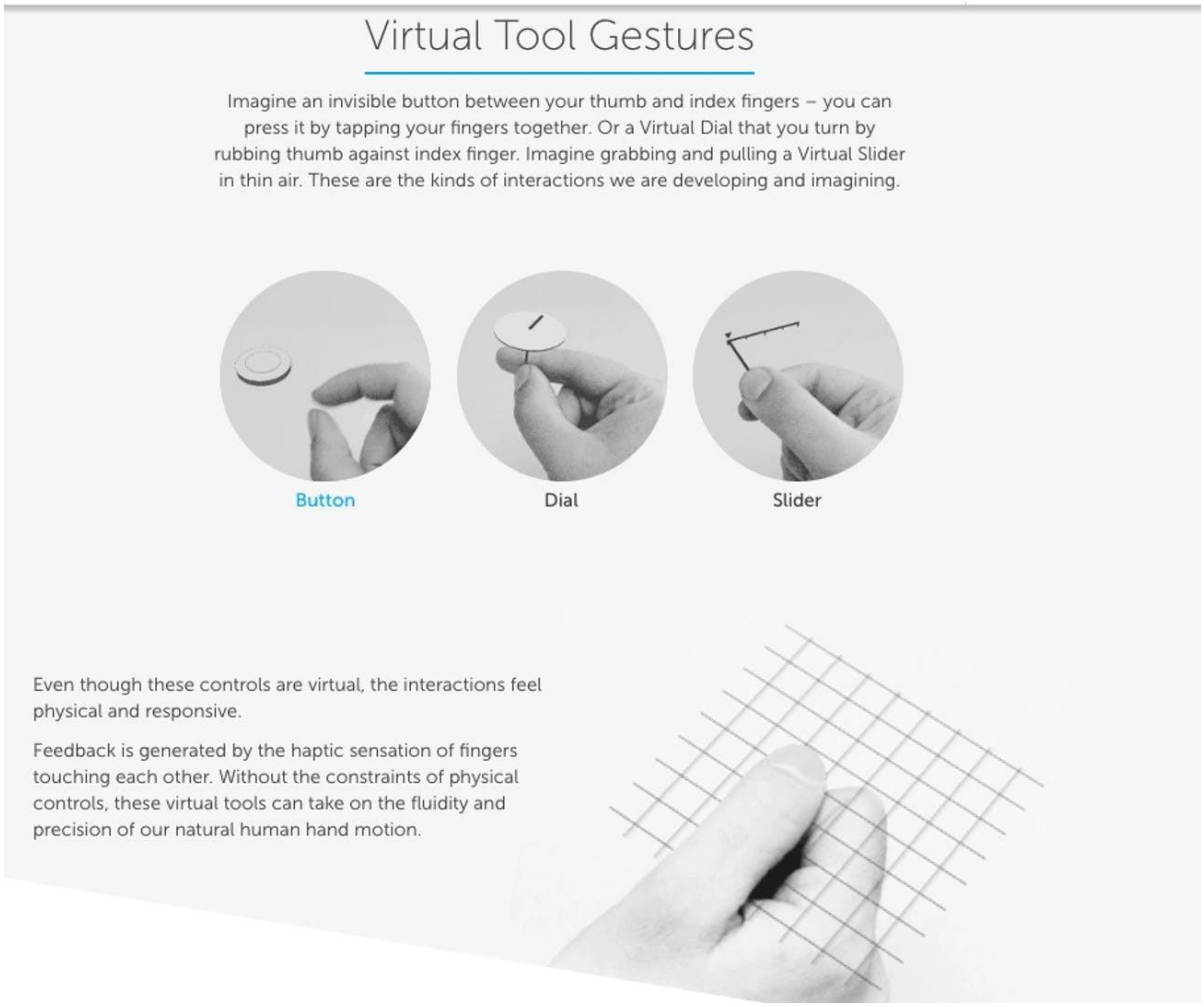
Results of the survey seem to highlight a couple of interesting conclusions. Volunteers found the touch-screen (blue bars) more reliable and slightly easier to use, and hand gestures interface (purple bars) more secure, less distracting and slightly more useful. Moreover, they found the hand gestures system a little bit more desirable and worth buying. These opinions are the same regardless the vehicle used (1st -2nd bar for each colour). A closer look to the comparison between both cars indicates that the prototype (1st bar) was perceived as more reliable, secure, useful and easy to use than the car simulator (2nd bar), for both interfacing systems. It is quite likely that these opinions were influenced by the fact that the prototype car was stopped and the drivers didn't have to pay real attention to the road, and also that the car simulator was used with the infrared illumination

**Consumer electronics sector :**

A wearable capture device is proposed to process the acceleration signals for gesture recognition applications. Capturing device consist a prototype system, which includes a gesturing sensing device and application program for mobile phone, is programmed to accomplish the gesture based real-time interaction. Sensing device consist an accelerometer, a microcontroller and Bluetooth component for realizing and gathering acceleration signals coming from hand gesture movement. LCD display is used for display coordinates axes. With the device worn on hand, user manipulates the smart phone by 5 predefined gestures.

The main objective of hand gesture study will be to create a technique that may identify specific human being movements and make use of them to express data as well as for gadget manage. Sensing and identifying gestures are two crucial issues to realize gestural user interfaces. The use of camera is an early developed technology to sense gestures, but it has not been applied in most mobile cases due to challenging problems such as changing light and background. Hence to overcome problems a prototype module has been proposed. A prototype is proposed to process acceleration signals for gesture recognition.






A wearable gesture-based interaction prototype demonstrates the feasibility of hand gesture interaction in mobile application that is based on the fusion of acceleration signals. A wear-able gesture capture device is designed to acquire acceleration signals, and an algorithm framework is proposed to realize gesture Classification on mobile devices. An interaction program is developed for the mobile device to realize gesture recognition and to manipulate the mobile device taking recognition results as instructions. Our prototype supports 8 gestures, a large gesture vocabulary for mobile device-based systems. The experimental results from interaction testing show that gesture-based interaction is feasible and performs better with experienced users although its efficiency needs further improvement.

**Health care :**

- Improving patient care
- Better understanding of diseases and symptoms
- Creating a virtual environment for medicine practitioners


In recent years, society is facing many sociological and financial challenges, so there is less time for the elderly or disabled persons. There is an increase in the number of disabled patients in society in which the patient cannot take care of themselves. A full time caretaker may be required to continuously monitor these patients, which is not always possible due to social or financial constraints. The existing electronic bed systems available in hospitals have only two movements of the bed (Up and Down). So, to minimize care takers requirement and increase the comfort level of the patients here, we have proposed an automatic bed position control system for disabled patients in which we have added two more movements to the existing electronic bed systems (Left and Right) movement of the bed based on different hand gestures. A particular gesture inputs given by the patient to the system, then this gesture is given to the micro-controller via RS-232

communication cable. The micro-controller further processes this input from the system and changes the position of the bed automatically with the help of DC motors and the accelerometer is used as a sensor to detect and display the patient fall.

Hll-Rom 305

Hill-Rom 80

Hill-Rom 405

Hill-Rom 900

1. Hill-Rom M: 305 :- The Hill-Rom 305 is an adjustable, manual hospital bed designed to deliver Hill-Rom quality and safety at an affordable price with easy-to-use features facilitate caregiver tasks.

2. Hill Rom 80:- The Hill-Rom-80 Extended Care Bed distributed by Hill-Rom offers residential comfort in a safe, reliable, and flexible solution at the right price.

3. Hill-Rom M: 405:- Side-rail design shields the patient, to help prevent patient falls. One step ahead board removal for rapid access to the patient's head

. 4. Hill-Rom M: 900:- The Hill-Rom 900 beds offer maximum functionality with minimal complexity, so you spend less time and energy operating the bed and more time focusing on essential tasks such as mobilizing patients and ensuring their safety.

EXPERIMENTAL RESULTS

In order to recognize the hand gesture recognition using the background subtraction algorithm, the accurate preprocessing operations are performed on current images, taken from the created database as well as the on-hand gesture images of people of varying age group.


This can be effectively used in real time processing in real time applications. Here we use different wavelet transform out of which Smelt wavelet transform give results accurate as compared with other transform. The Silverware let transform gives an accuracy of 96%. The gesture is then given to the micro-controller via RS-232 communication cable for further processing, the controller processes the transferred data code gesture and passes this to DC motor driver to drive the motor clockwise or anti-clockwise depend on the gesture given by the patient the position of the bed changes as per the given gesture input as up/down and right/left.

the hand gesture recognition is carried out by using background subtraction algorithm. Background subtraction is a common method of motion detection. It is a technique used for the detection of difference between the background image and the current image. There are different tools for gesture recognition, based on the approaches ranging from statistical modeling, computer vision and pattern recognition, image processing, connection systems.

———————————*THE END*———————————