Classification of Wall - Following Robot Navigation

1.0 Description of features and Classes in Study Set

There is a total of 5,456 instances and 24 features in this set. All features are numerical and real values. Each feature is a different ultrasound reading at different reference angles, where there are two actual sensors placed in the front and the back of the robot. Each reference angle value will influence where the next movement of the robot will be. Each of the sensor reading is sampled at a rate of 9 seconds and is collected simultaneously. Each sample is the number of instances. Below will be a table with a detail description of the specific reference angle for each feature. Further, there is 4 classes (Move Forward, Slight Right Turn, Slight Left Turn, Sharp Right Turn) which describes the movement the robot will make based on the sensors (Features).

ULTRASOUND	REFERNCE ANGLE OF ULTRASOUND READING
US1	FRONT OF THE ROBOT 180
US2	-165
US3	-150
US4	-135
US5	-120
US6	-105
US7	-90
US8	-75
US9	-60
US10	-45
US11	-30
US12	-15
US13	BACK OF THE ROBOT 0
US14	15
US15	30
US16	45
US17	60
US18	75
US19	90
US20	105
US21	120
US22	135
US23	150
US24	165

1.1 Description of study set and why classification task

This study set contains data from a wall - following robot used for simple navigation, where the goal is to predict the position of a robot based on the sensor readings from the robot during its movement. The data is

collected is course of a robot navigating through a room following a wall in a clockwise direction. The data set is also non- linear in nature. The field of robotics is a versatile field, to understand and to be able to predict robot movement can play an important role in a range of real-world tasks. For example, robots are used in aerospace programs like the popular Mars rovers or even in underwater exploration.
Using classification tasks, we can predict which direction the robot will navigate by using new sensor data. One method of classification that will be used is the K- Nearest Neighbor algorithm (KNN), where it is supervised method and uses previous labeled points and distance to classify new points. Using KNN is efficient because the data set is quite large having around 5456 instances

Preliminary Treatment

Eliminating the rows/cases missing feature values (no missing rows/ cases so eliminated nothing), then roughly rebalance the size of the classes by cloning a small class to double or triple its set size. The imbalance of classes can cause biases in the model, thus needing to rebalance the classes is important. This was done on excel, where a class of 328 of Slight - left turn was multiplied 6.85 times leading to a total of 2205 Slight left Turn. For Slight right turn the class 826 was multiplied 2.66 times to have an output total of 2205. The final total of instances is 8757 with 24 features. It is probably not best to replicate a small class so many times, which can lead to a reduced level of accuracy in the prediction. After rebalancing all classes, the data set was then standardized for each feature. We want to standardize the data where the mean = 0 and with a standard deviation of 1. This will prevent biases between features.

BEFORE CLONING

CLASSES	TURN
Move- Forward	2205
Sharp - Right - Turn	2097
Slight- Left- Turn	328
Slight - Right - Turn	826
Total	**5456**

AFTER CLONING

CLASSES	TURN
Move- Forward	2205
Sharp - Right - Turn	2097
Slight- Left- Turn	2205
Slight - Right - Turn	2205
Total	**8712**

1.3 Evaluating the Discriminating power of Features

Optimizing the performance of a classifier requires prior knowledge of maximum achievable accuracy using a particular set of features. It is desirable to know that the set of features is separable, a histogram test will be done to compare classes on each feature. I will do a histogram on all four classes on three fixed features to visually differentiate which features are separable between classes.

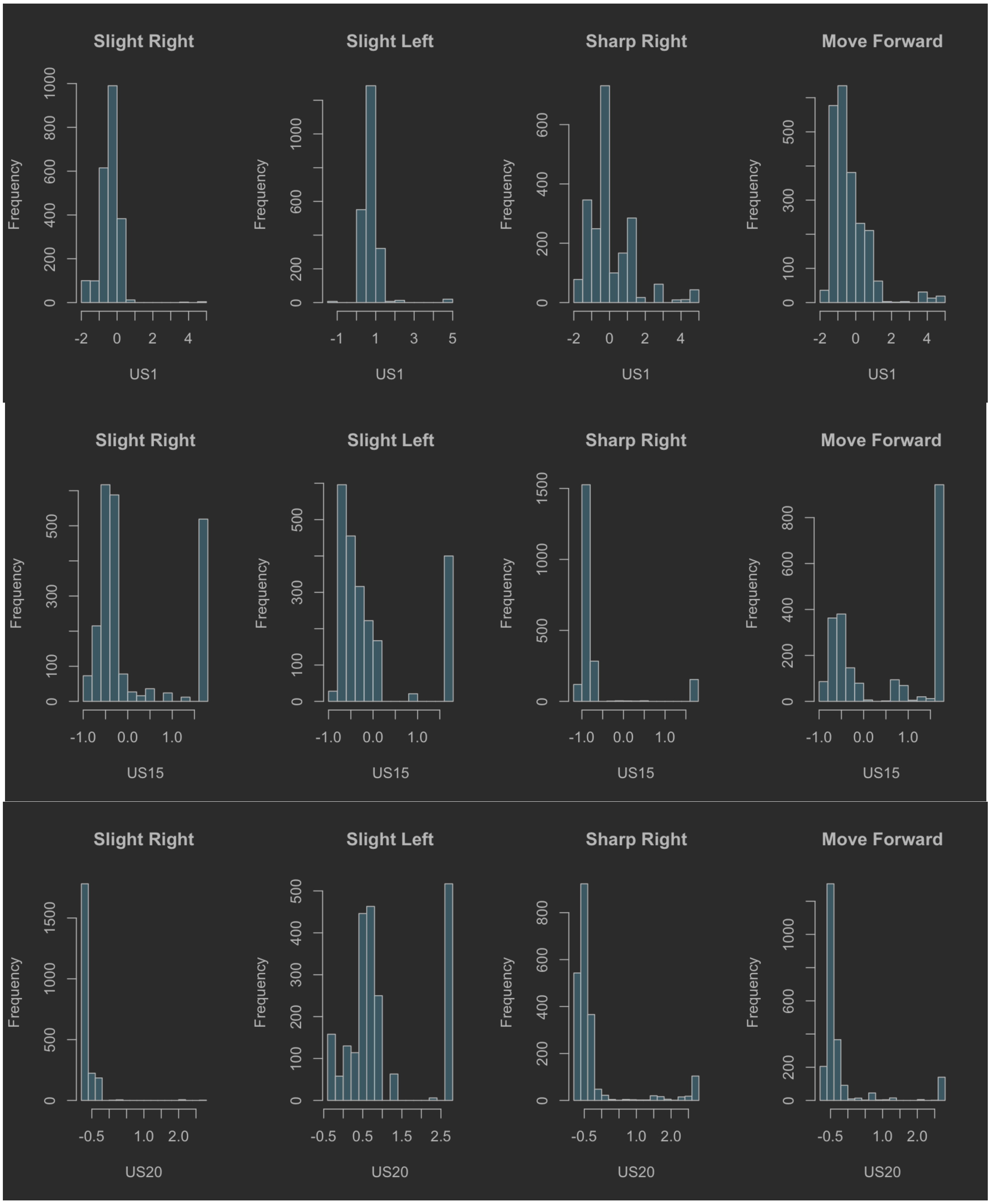

Figure 1 - Histogram across classes of MoveFoward, SharpRight Turn, SlightLeftTurn, and SlightRightTurn against US1,US15 and US20.

Each figure is labeled with its respected class at the top, with the x axis as the ultrasound sensor and the y axis as the frequency.

Based on the histograms of three different features, we can see that US1 has very similar distribution which will lead to a lower accuracy rate. The prediction model will have trouble because each distribution is not distinct. Taking out this feature in the future will increase accuracy. However, in US15 and US20 we can see that the distribution is separable and distinct in each class. However, between features they are separable as well.

1.4 KS – Test to confirm distribution

The KS – test uses discriminatory power of the model in separating the classes, where the highest separation is the cumulative distance between the two classes. The higher the KS distance value the more separation between the classes. Having separation will help the prediction model separate and classify the classes correctly. Below shows that in each feature we can have high separation (not US15), leading to features being separable.

US1: Slight_Right and Slight_Left
D = 0.87982, p-value < 2.2e-16
US5: Slight_Right and Slight_Left
D = 0.58141, p-value < 2.2e-16
US15: Slight_Right and Slight_Left
D = 0.21814, p-value < 2.2e-16
US20: Slight_Right and Slight_Left
D = 0.9932, p-value < 2.2e-16

2.0 The K- Nearest Neighbor Algorithm

The KNN algorithm is a supervised nonparametric learning model, where the relationship between features and targets are not assessed solely on a fixed number of parameters. The KNN method uses Euclidean distance or Manhattan distance (better for when you have redundant features) to define similarity in features. For example, the features are identical the similarity distance should be zero. The distance we choose is called the k value, where we not only want to choose the nearest neighbor, but also consider a small number of nearby neighbors. Having a range of neighbors will give us different perspective and reduce the noise amount in the data. Further, the smaller k parameter, the more the algorithm will adapt to the data, which is overfitting. The larger the k parameter it causes it to be more abstract from the real data, leading to accounting for irrelevant classification / examples. To create a KNN algorithm there are four parts that make up the algorithm.

1. A matrix that contains the predictors associated with the training data
2. A matrix containing the predictor associated with the data we wish to make predictions on
3. A vector containing the class labels for the training observations
4. A value for K used to be used by the classifier

2.1 Preparing a Training and Test Sets

Before creating the train and test sets, the target class will be split up into 4 subsets by each class. Then each subset will then be split arbitrarily where the Train set will be 80% of cases and the Test set will be 20% of cases in the subset. After each subset is made from each class, they will be unionized to make a full Train and full Test set. This is to keep the classes balanced throughout the Train and Test set to reduce bias.

2.2 Applying the K-NN algorithm with multiple k

To apply the KNN algorithm, a matrix is created for the training set and test without the class predictors which will be labeled Train_robot_no and Test_Robot_no. Then the two vectors are created that contain the class labels for the training observation and testing observation which are labeled TRAIN_Robot_label and TEST_Robot_label. We then apply the knn() function on the training and test set to predict the positional movement based on the 24 sensor features. Further, a set seed function is applied to prevent overlapping observations tied to the nearest neighbor. The knn function ran will test K values from a range of 5 to 100 in segments of fives. Testing a large range of k values will aid in understanding the fit of the model. Below will list the accuracy rates of the testset and the accuracy plot for both the train and test set. The plot of the k values will help us visualize if the K value will overfit between the train and test set.

TRAIN SET	
K VALUE	ACCURACY
5	92.17965
10	89.49634
15	87.97532
20	86.56909
25	85.66509
30	84.74674
35	84.01492
40	83.26876
45	81.99168
50	80.91548
55	80.12627
60	79.86799
65	79.32271
70	77.95954
75	77.02683
80	77.15598
85	76.09413
90	75.56321
95	75.39102
100	74.8314

TEST SET	
K VALUE	ACCURACY
5	89.214
10	87.32071
15	85.8864
20	85.25531
25	83.70625
30	83.36202
35	82.67355
40	81.58348
45	80.0918
50	79.11647
55	78.65749
60	78.60011
65	77.73953
70	76.99369
75	76.24785
80	75.78887
85	75.1004
90	74.87091
95	74.23982
100	74.23982

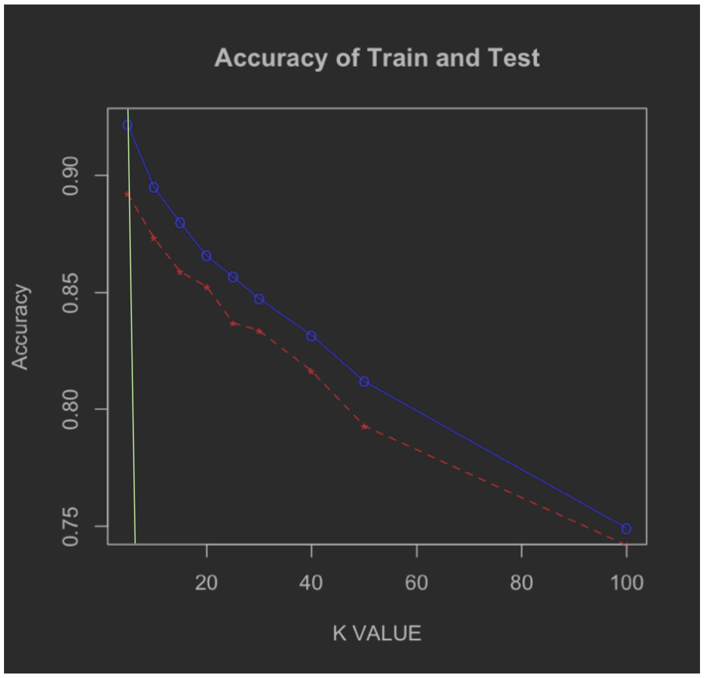

Figure 2: the comparison of accuracy between test and train set.
The X axis of the graph shows the K values from 0 to 100 and Y axis show the accuracy rate where 0.90 is 90%. The graph shows two lines; Blue (Train set) and Red (Test set). Each dot indicates the correlated k value and accuracy point. The green vertical line indicates the highest accuracy which is below 20.

Based on both the accuracy tables of train and test set we can see that the accuracy rate is significantly higher in the train set. This is expected as the training set is being trained on itself. Further comparing the two tables I can determine that the best k- value is going to be K = 5. We can see that accuracy decreases as K increases. Using figure 1, we can also determine that there is a no overfitting in the model. Leading to a final decision of K =5 as the "best" k value.

2.3 Using the "best" k value and evaluating the corresponding confusion matrixes

Using the "best" k value as 5, the KNN() function will be run on both the test and train set. After receiving the accuracy values, a confusion matrix will be created on both sets. The corresponding accuracy rates will also be listed below.

		Prediction Test Robot Label			
	Test K = 5	**Move Forward**	**Sharp Right**	**Slight Left**	**Slight Right**
TRUE	**Move Forward**	86%	9%	0%	5%
	Sharp Right	5%	93%	0%	2%
	Slight Left	4%	4%	92%	0%
	Slight Right	10%	5%	0%	86%

		Prediction Train Robot Label			
	Test K = 5	**Move Forward**	**Sharp Right**	**Slight Left**	**Slight Right**
TRUE	**Move Forward**	93%	6%	0%	2%
	Sharp Right	6%	93%	0%	1%
	Slight Left	3%	3%	94%	0%
	Slight Right	7%	3%	0%	89%

Total Test Accuracy :89.214%
Total Train Accuracy: 92.17965%

Test set confusion matrix
The confusion matrix on the left is the accuracy rate for the test set. We can see that 86% of move forward is classified correctly where generally 19% is a false positive and the false negative is 14%. For the accuracy of Sharp Right, we can see that 93% of the values are classified correctly. Further, 18% is a false positive and false negative is 7%. Slight left had an accuracy of 92% with 8% false negatives and 0% false positives. Lastly Slight right had an accuracy rate of 86% with a 15% false negative and 7% false positive. Overall in this confusion matrix, we can conclude that Slight left will have the highest accuracy because it has no misclassification in false negatives.

Train set confusion matrix
The confusion matrix on the left is the accuracy rate for the test set. We can see that 93% of move forward is classified correctly where generally 16% is a false positive and the false negative is 7%. For the accuracy of Sharp Right, we can see that 93% of the values are classified correctly. Further, 12% is a false positive and false negative is 7%. Slight left had an accuracy of 94% with 6% false negatives and 0% false positives. Lastly Slight right had an accuracy rate of 89% with a 10% false negative and 3% false positive. Generally, we can see that accuracy rates are higher in the train set, as this is expected. The model is being tested on the training set. This will lead to lower false positives and false negatives.

4.0 - Increasing accuracy for the KNN test by taking out classes.

The accuracy rate total without altering the feature and classes is 0.89214 for the test set. In an attempt to increase accuracy, I will discard the Slight Left turn. The reason to discard this is class is because cloning a small class to almost 6 times can cause inaccuracies. However, from the previous confusion matrix we can assume that the inaccuracies are very small. But taking out the smaller class may still improve accuracy. Apply KNN algorithm where K =5.

. Total Accuracy
91.60423%

TRAIN Set Prediction Without Slight Left

TRUE	Test K = 5	Move Forward	Sharp Right	Slight Right
	Move Forward	93%	6%	2%
	Sharp Right	6%	93%	1%
	Slight Right	7%	4%	89%

Total Accuracy
89.40092%

TEST Set Prediction Without Slight Left

TRUE	Test K = 5	Move Forward	Sharp Right	Slight Right
	Move Forward	90%	8%	3%
	Sharp Right	8%	90%	2%
	Slight Right	9%	2%	89%

Comparing this confusion matrix from the previous one, we can see that total accuracy decreased for the trainset. However, there was a slight increase in total accuracy in the test set of about .200%. We can also see that the overall move forward accuracy and slight right increased in accuracy. The Move forward has a smaller false negative accuracy of 11%. All the false positive and false negatives have lower accuracy in both the train and test set. As this is expected since one class has been taken out. In the train class the accuracies, it has stayed relatively the same however, slight right classified as move forward increased by 1%. This probably caused the accuracy to decrease in the overall model.

Overall by taking out one class, the model barely increased in accuracy and had less false negatives and false positives in the TEST set.

4.1 Increasing accuracy AGAIN by decreasing features by using Fisher LDA

After applying the KNN algorithm (K=5) to three classes, a fisher score test will be done to see the importance of features in the data set. The Fisher LDA test is a technique that does dimension reduction while preserving as much as the discriminatory information as possible. It uses maximum distance between projected class means and minimize projected class variance to achieve the fisher score. I will discard the 11 least important features based on the fisher score (score<500 will be discarded). Discarding the less important features will increase accuracy as those features could possible cause the prediction model to classify false positive or false negative. A KNN algorithm (k=5) will then be done on THREE classes and the new 13 features. The confusion matrixes will be added below. The features taken out will be US(8,22,7,6,9,10,1,5,12,11,13).

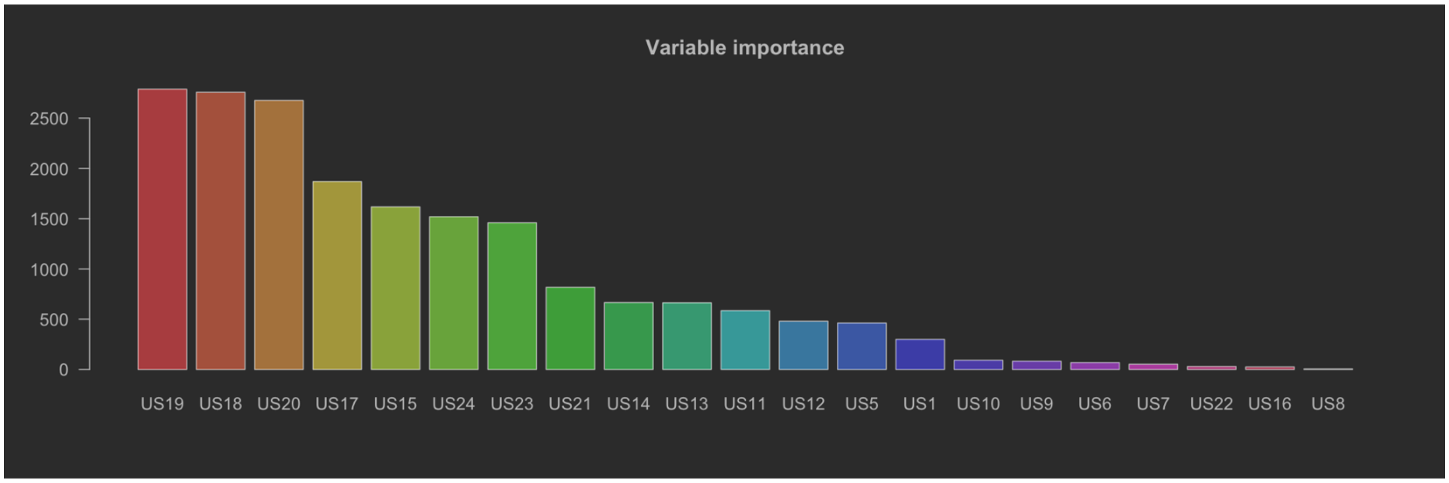

Figure 3: the importance of each variable across 24 features
The y- axis is the fisher score measuring the importance by means. The x- axis is the corresponding features.

Total Accuracy
89.789%

Test Set Prediction Without Slight Left & features

	Test K = 5	Move Forward	Sharp Right	Slight Right
TRUE	Move Forward	88%	6%	5%
	Sharp Right	8%	91%	0%
	Slight Right	7%	3%	90%

Total Accuracy
92.28%

Train Set Prediction Without Slight Left & features

	Test K = 5	Move Forward	Sharp Right	Slight Right
TRUE	Move Forward	94%	4%	2%
	Sharp Right	6%	93%	1%
	Slight Right	6%	3%	92%

Based total accuracy, we can see that taking out the features has improved the total accuracy for both test and train set. Although the accuracies are only increasing slight, the improvement means that the features have different importance' by mean. We can also see there is a lower number of false positives and false negatives in both train and test set. We can assume that not all features have equal importance in classification.

4.2 Implementing PCA on KNN with all FOUR classes

Principal Component Analysis uses the sum of percentage of variation in the data set to see how strongly correlated they are with one another. It uses eigenvalues and eigenvectors to create the number of dimensions in the dataset. Using a PCA requires all numerical features, as it is using its correlation values, eigenvalues, and eigenvectors. Unlike LDA, the higher variance the more important the features are. Using PCA reduction will increase accuracy on KNN by decreasing the number of features (high dimensionality to risk over fitting). PCA uses an algorithm that linearly transforms dimensional input where $r < p$ to discard/ minimize the features with less variance. First, we must look at the correlation matrix pixels to visualize if the features are correlated with each other. If they have high correlation it can indicate a redundancy in features, where they have potential to have dimension reduction. After doing the PCA, the eigenvalues and eigen vectors will be extracted to find variance percentage at 95% (in this set it will be at dimension 20). Here we can see which dimension is 95% of the data set and to better visualize this, a scree plot will be done. Lastly, a contribution percentage of features will be plotted, and the top values will be exhibited. The values under the dimension mark will be discarded in a new train and test set (the values discarded will be US[5,10,11,9,8,6,7,22,4]) . This will create a new data set having the top variances in the 95% dimension only which should improve accuracy with KNN (k=5) is applied. The KNN algorithm will be done on the new features (high variance) and all four classes.

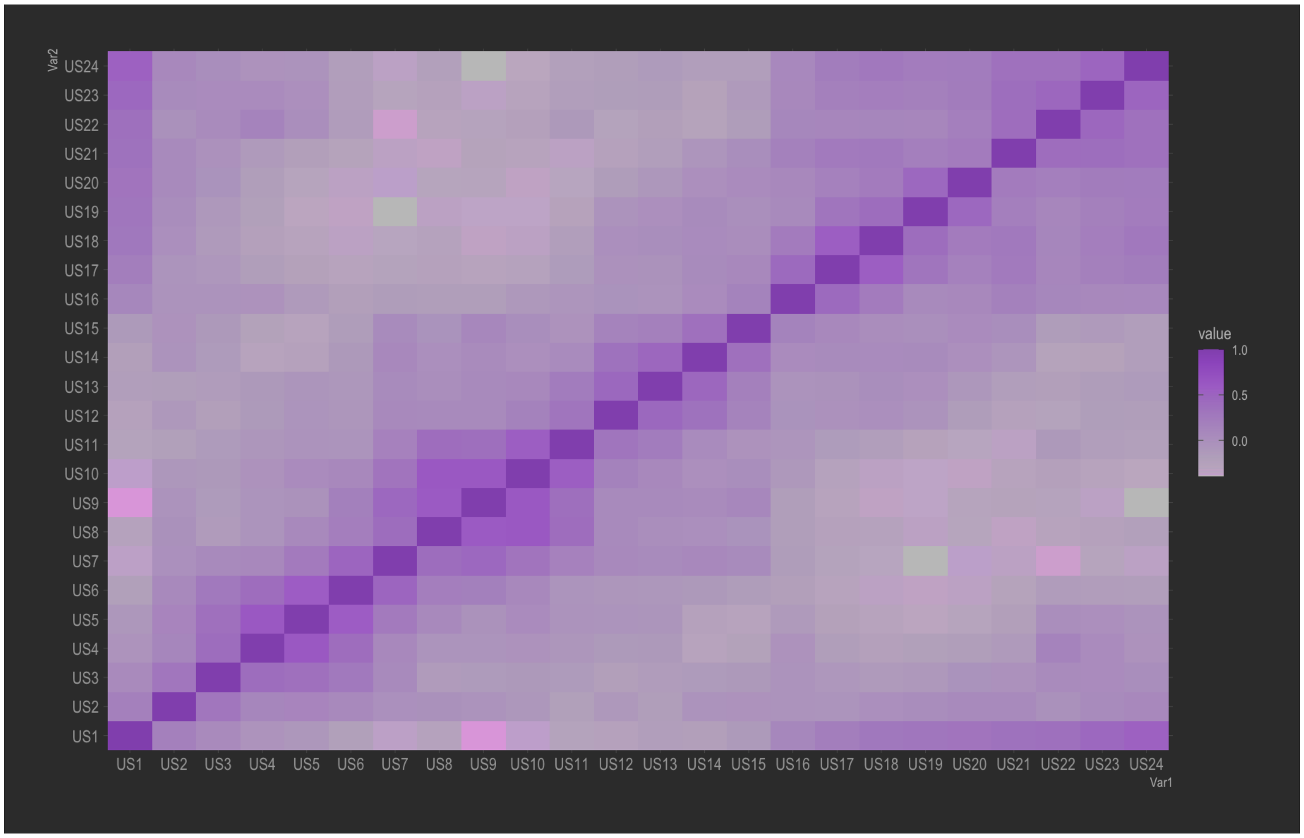

Figure 4 – Correlation pixel heat map with all 24 features

The x and y axis are the features, the legend on the side exhibits the correlation scale. The darker the purple the more correlated the features are with each other.

Based on the pixel plot, there are some features that are highly correlated with each other, but the majority features are in the medium range. Those that are highly correlated should be extracted as they could indicate redundancy of features.

	eigenvalue	Variance percent	Cumulative variance percent
Dim.1	5.6708189	23.628412	23.62841
Dim.2	3.3792349	14.080145	37.70856
Dim.3	1.8177181	7.573825	45.28238
Dim.4	1.4060637	5.858599	51.14098
Dim.5	1.2938336	5.390973	56.53195
Dim.6	1.069201	4.455004	60.98696
Dim.7	1.019029	4.245954	65.23291
Dim.8	0.9377399	3.90725	69.14016
Dim.9	0.689336	2.872233	72.0124
Dim.10	0.6377125	2.657135	74.66953
Dim.11	0.6274297	2.61429	77.28382
Dim.12	0.6098026	2.540844	79.82467
Dim.13	0.5374065	2.239194	82.06386
Dim.14	0.4881027	2.033761	84.09762
Dim.15	0.4738946	1.974561	86.07218
Dim.16	0.4610678	1.921116	87.9933
Dim.17	0.4369509	1.820629	89.81393
Dim.18	0.4143313	1.726381	91.54031
Dim.19	0.3947955	1.644981	93.18529
Dim.20	0.3776501	1.573542	94.75883
Dim.21	0.3626469	1.511029	96.26986
Dim.22	0.3226514	1.344381	97.61424
Dim.23	0.3046099	1.269208	98.88345
Dim.24	0.2679724	1.116552	100

This table shows the values, variance percentage and cumulative variance percentage. Ideally, we want to have around 95% variance, thus we can conclude that we should use up to dim 20 to find the highest variance features.

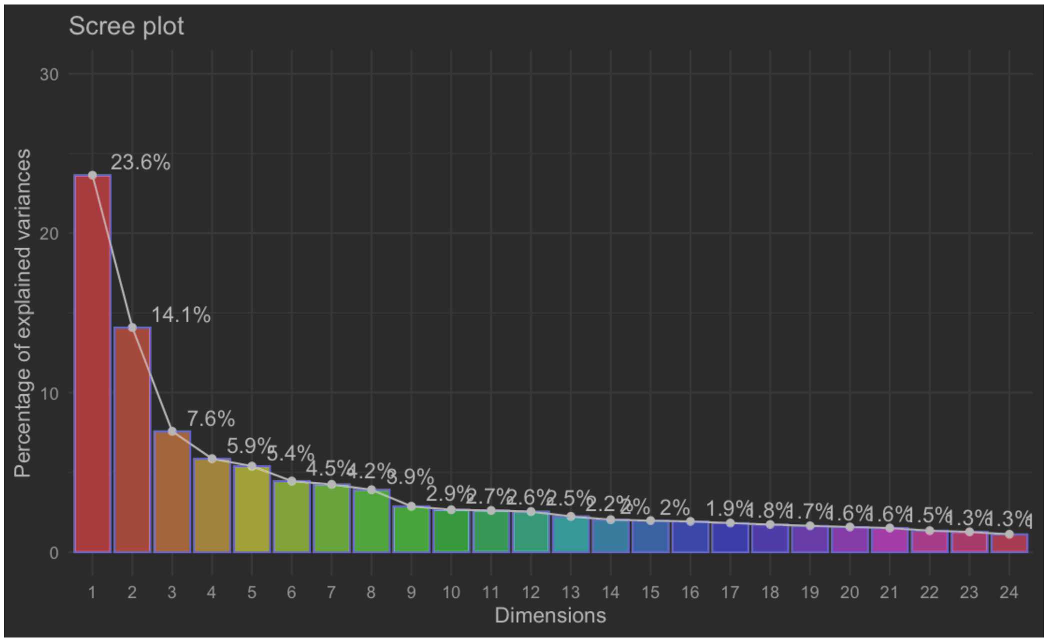

Figure 5 – A scree plot showing the percentage of variance X- axis is each dimension, y- axis is % variance.

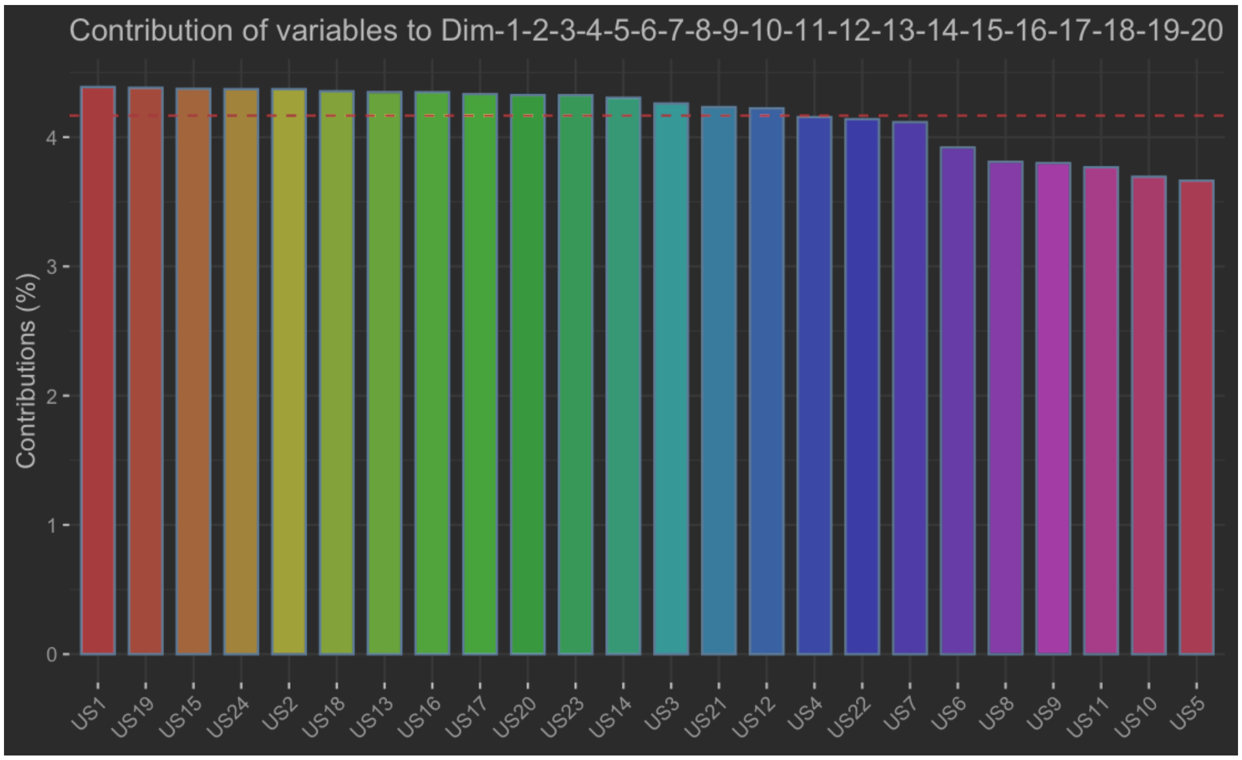

Figure 6 – contribution plot of dimensions with features
The y- axis being the features and the x- axis is the contribution percentage (0 being the lowest). This plot looks at the highest importance by variance of 95%. The dotted line shows where the 95%-mark ends, any feature underneath the red dotted line will be discarded.

A total of 9 features will be discarded leaving to a total of 15 features. These 15 features will be applied to a new KNN to test to see if feature variance importance influence the accuracy of the algorithm. The reduction of features from 24 to 15.

Total Accuracy
93.528%

		Prediction Train Robot Label with PCA			
	Test K = 5	Move Forward	Sharp Right	Slight Left	Slight Right
TRUE	Move Forward	93%	4%	0%	2%
	Sharp Right	6%	94%	0%	0%
	Slight Left	4%	2%	94%	0%
	Slight Right	5%	2%	0%	93%

Total Accuracy
90.648%

		Prediction Test Robot Label with PCA			
	Test K = 5	Move Forward	Sharp Right	Slight Left	Slight Right
TRUE	Move Forward	87%	8%	0%	5%
	Sharp Right	8%	92%	0%	0%
	Slight Left	5%	2%	93%	0%
	Slight Right	8%	2%	0%	90%

After running the KNN, we can see that the total accuracy increased a around 1% in the test set compared the ordinary knn with all features. The number of false negatives and false positives have also decreased in both train and test set. We can also see that train accuracy is higher test, thus this is not overfitting in the models. Comparatives the LDA, the PCA has higher accuracy as well even with the highly cloned class. To increase accuracy even more taking out the slight left class will increase the accuracy as it has been cloned too many times. The PCA test confirms that not all features have equal importance in classification.

Warning errors:
I was unable to do the weighted KNN due to and error saying, “too many ties” This may be caused by too many points equal distance from the point I am trying to classify. I attempted to alter the k – value to see if this fixed the problem however it did not. I also attempted the KS.TEST way and it gave me the same error. Thus, leading me to attempt a PCA instead, which worked fine.