

# Mexican Sign Language recognition using hand-landmarking and a KNN algorithm

Diego Santa Cruz<sup>1</sup>

<sup>1</sup>Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO), Guadalajara, Mexico

Corresponding author: Diego Santa Cruz (e-mail: adrian.santacruz@iteso.mx).

**ABSTRACT** The purpose of this project is to create and evaluate a data preparation process for a future project centered on developing a Mexican Sign Language classifier. We obtain 3-dimensional coordinates for hand landmarks from images of Mexican Sign Language alphanumerical signs obtained using Google's MediaPipe module. These coordinates are then transformed to wrist-centered, orientation-independent representations.. A statistical analysis assesses the data's quality, followed by dimensionality reduction using Principal Component Analysis (PCA). Finally, the K-Nearest Neighbors (KNN) algorithm is applied to evaluate both the dataset's suitability and the algorithm's applicability.

**INDEX TERMS** Hand landmarking, K Nearest Neighbors, Sign Language Recognition

## I. INTRODUCTION

Around 300,000 people use Mexican Sign Language (MSL) as a means of communication. This represents around 13% of the hard-of-hearing population in Mexico and 0.2% of the general Mexican population. These low numbers illustrate the isolation that hard-of-hearing people live in. In part, this can be explained by the low awareness and interest in this language and the lack of access to its teaching. It is therefore important to find new ways to teach MSL to both speaking and non-speaking people in Mexico that are easy to access.

Worldwide, a popular solution to address this problem is the development of Sign Language Recognition Systems [1]. Among these, a few have focused on MSL [2], [3]. We are developing this project in the interest of contributing to this research.

## II. DATA ACQUISITION AND PREPROCESSING

### A. SOURCE DATA ACQUISITION

We will be using the "Mexican Sign Language's Dactylology and Ten First Numbers - Labeled images and videos" dataset [4] [5].

It is a set of videos where 11 different volunteers perform 39 alphanumerical MSL signs. Specifically, this includes the signs corresponding to the 29 letters of the MSL alphabet (which include the 26 contained in the English alphabet as well as "LL", "RR" and "Ñ") and the signs for the numbers 1 through 10. Each sign is performed by each volunteer 10

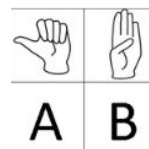
times, 5 times with each hand. This means the dataset contains around 4290 videos in which an isolated sign is performed.

The background in the video as well as the volunteers' clothing and their distance to the camera vary, which should aid in making the model trained on this data more robust. Some of the signs are static (meaning there is no movement involved in the action) and some are dynamic (meaning movement is part of the sign).

The videos are filed in the following manner:





- First, per volunteer, for example "person number 9"
- Second, per number repetition and dominant hand used for the signing, for example "Left Hand repetition number 2" or "Right hand repetition number 5"
- Finally, per sign being performed, for example "letter C"

In future projects, we intend to use all 39 classes. However, for this project, we only used two classes: "A" and "B". Here is a schematic representation:



It is worth noting that choosing these two classes simplifies the problem quite a bit.

First of all, one thing to consider is the difference between static and dynamic signs. In Mexican Sign Language, as in many sign languages, some signs require movement whereas others do not. For example:

- The signs for “L” and “LL” are identical in configuration (meaning the shape the hand is in), but are different in movement:
  - The sign for “L” is static. Here is a schematic representation:
 
  - The sign for “LL” requires moving the hand from one side of the body to the other. Here is a schematic representation:
 
- The signs for “K” and “P” are also identical in configuration, but different in movement:
  - The sign for “P” is static. Here is a schematic representation:
 
  - The sign for “K”, while keeping basically the same position with regards to the signer’s body, requires rotating/waving the hand at the wrist’s level. Here is a schematic representation:
 
- The signs for “1” and “100” start out the same, but differ throughout the sign’s execution:
  - The sign for “1” is static
  - The sign for “100” starts out the same as the sign for “1”, and then moves from one side of the signer’s body to the other while folding the index finger (change in hand configuration)
  - See page 469 of [6] for visual representations

The signs we chose for this project (“A” and “B”) are both static signs.

Another complexity to consider is the fact that some signs use both hands simultaneously. For example, the sign for “animal” [7] is a dynamic sign in which one of the two hands is placed in a fist, and the other hand moves in a circle around that fist (note: the second hand also changes in configuration during this circular movement).

The signs we chose for this project both use only one hand.

Another intricacy to mention is the fact that for some signs the relation of the position of the hand (or hands) with specific body parts is relevant. For example, the signs for “father” and “mother” [8] require placing the hand near the chin in the configurations for the letters “P” and “M”, respectively.

Also, particular spaces can be used for grammatical purposes at times.

The signs we chose don’t have a specific area in which they have to be performed, although it generally is done in front of the chest.

In addition, some signs make use of facial expressions to communicate meaning. For example, the sign for “fat”, in addition to the action performed by the hands (and arms even), includes “puffing your cheeks out”. The sign for “skinny”, in addition to the action performed by one of the hands, requires “sucking your cheeks in”.

The signs we chose for this project generally have no associated facial expressions.

Additionally, another challenge which some Sign Language Recognition projects try to take on is the continuous recognition of signs. For example, some projects focus on fingerspelling, which is the act of performing a sequence of alphabetical signs to spell out a word in another language like Spanish or English. By taking a video which corresponds to a whole sequence of signs which are not separated instead of having one sign per video, additional difficulties appear.

It is important to note that fingerspelling is not very representative of a signer’s communication experience. They are sometimes used to communicate with non-signers, for acronyms or in cases where no word in the signed language exists for a word in a spoken language. However, most of the communication occurs with signs that correspond to concepts of nouns, verbs, adjectives, and such. Continuous Sign Language in this case is also an area of study, where sometimes the language’s grammar can be used to determine the signs being performed.

For this project we focus on isolated, alphanumeric signs only.

Finally, it is important to note that there are many variations in Mexican Sign Language. Some of the signs for concepts mentioned here (for example for “100”) have many different versions. For a visual reference that includes many variations for different concepts, see [6].

## B. DATA PREPARATION

### 1) FRAME EXTRACTION

We have around 110 videos for each of the classes. Considering this and after analyzing the videos in the source data set, we decided to extract the last 15 frames in from each of the videos using the OpenCV python library [9].

Note that it is at this point that we are deciding to treat only static, isolated signs. From this point on we are treating each frame separately and remembering only what sign the video was labeled as, but not which video each frame actually came from.

## 2) HAND LANDMARK EXTRACTION

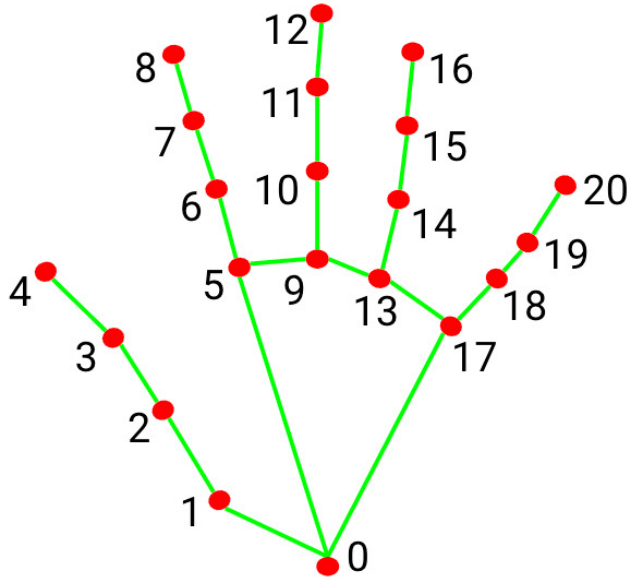


Figure 1:  
Hand Landmarks Identified by Google's MediaPipe [10] module.

For each of the extracted frames, we use Google's MediaPipe hand-landmarking model [10], [11] to extract 3-dimensional coordinates for 21 specific points of the hand which can be seen in figure 1.

It is worth noting that up to here we are following the same steps used in [3], the paper which was originally developed to make use of our datasets [4] [5].

Note that MediaPipe also offers pose detection and face landmarking detection. In future projects in which we consider signs for which the hands' positions with relation to specific body parts are relevant, those MediaPipe models could be useful to extract coordinates for specific points of the face, torso and arms.

Also, we are making use of the fact that the signs in our dataset are performed with only one hand, and that the hand being used is labeled, to extract the information for that hand only. However, MediaPipe's hand landmarking model can extract this data for multiple hands, so it is possible to envision expanding the methods used for this project to recognize signs which use both hands.

## 3) DATA TRANSFORMATION

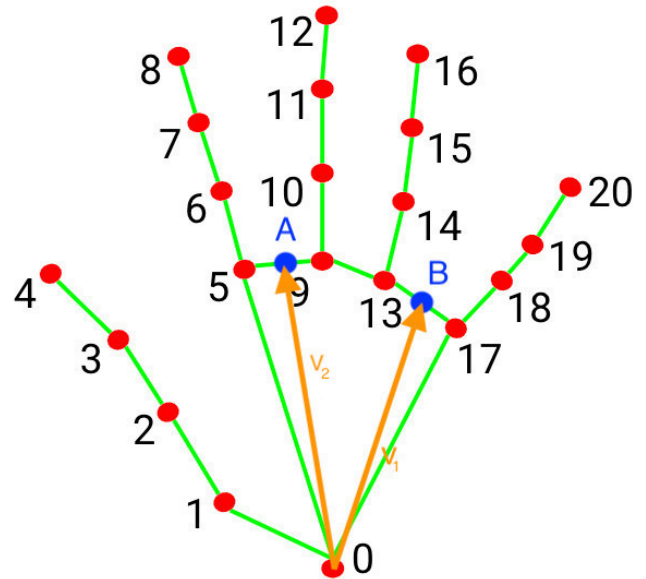


Figure 2:  
Vectors  $v_1$  and  $v_2$  for the new base superposed on MediaPipe's [10] hand landmarks

In the interest of obtaining more meaningful data, the idea for this project was to perform a series of transformations on our 21 coordinate triplets. We obtain data corresponding to the following:

- The center of mass of the hand
  - Because the classes we chose in this project are not impacted by the hand's position with regards to other body parts, this information will likely not be very useful when training our model.
  - Nevertheless, in the future we might compare the coordinates for the center of mass of the hand to the coordinates of other body parts (such as the shoulders, chin, or ears) which we can extract using other MediaPipe models
    - See the comparison of "L" vs "LL" earlier
- The vector normal to a plane which approximates the palm of the hand
  - To do so, we first get the coordinates of two vectors which belong to the plane that approximates the palm
    - We choose  $v_1$  and  $v_2$  as in Figure 2, where A is the point between points #5 and #9, and B is the point between points #13 and #19
  - Then we obtain the vector normal to the plane by performing the cross product of  $v_1$  and  $v_2$ 
    - It is worth noting that we decided that we always  $v_3$  to "point out of the front" of the hand. Because of

this, the order of  $v_1$  and  $v_2$  will depend on whether we are using the left or right hand.

- Fortunately, the videos included labels for the hand which is performing the sign. We can use this information when performing these transformations. Therefore, we are always able to order  $v_1$  and  $v_2$  in the desired way in the cross product.
- Because the classes we chose in this project don't have much important information in terms of the orientation of the hand, this data will likely not be very useful to train our model
- Nevertheless, in the future it could be interesting for cases with dynamic signs
  - See the comparison of "K" vs "P" earlier
- Also, we needed these transformations for an additional transformation
- For each of the 21 hand landmarks, we first transform their coordinates so that they are centered on the wrist (point #0)
  - This is so that regardless of where the camera might have been placed with relation to the signer or other such factors which have no impact on the sign, the coordinates will provide meaningful information
  - Remember that we also obtained the coordinates for the center of mass of the hand which we will be keeping, so we don't completely lose track of this information
- Then we apply a change a base from the natural cartesian base captured by the camera to the base  $(v_1, v_2, v_3)$ .
  - This makes the coordinate be more "hand-centric" and not based on arbitrary decisions of camera placement.
  - Also, small differences with regards to the angle in which the palm is pointing which often have no meaning should be less impactful in this new base
  - Remember that we also obtained the coordinates for the vector "pointing out of the palm" which we will be keeping, so we don't completely lose track of this information

After these transformations, we have the following information about each of our frames:

- i. 63 columns with the coordinates of each landmark with the wrist as the origin of the frame of reference

and  $(v_1, v_2, v_3)$  as the base. This gives us information about the "hand configuration" in a position-independent and orientation-independent way.

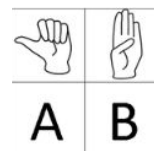
- ii. 3 columns with coordinates of the vector normal to the palm of the hand and 3 columns with the coordinates of the center of mass of the hand. Both coordinate triplets are in the original frame of reference. This gives information about the position and orientation of the hand. As mentioned earlier, this information is likely to not be very useful for the classes in this project but could be important for dynamic signs.
- iii. 1 column which indicates the class in a numeric form format.

We use the Apache Spark framework [12] to perform these transformations in a parallelized way.

#### 4) VARIABLE ASSOCIATION ANALYSIS

We use the Maximal Information Coefficient metric to evaluate the association between each pair of (non-constant) variables in our transformed dataset. See Figure 3 in the Additional Figures section for the full results.

The variable with the strongest association to our class turned out to be the y coordinate of the thumb's tip in the new coordinate system. Recall the hand configurations of our two signs:



We can see that the thumb's tip's position changes a lot with relation to the rest of the hand. Therefore the fact that one of it's coordinates has a strong association with the class indicates that this information has been kept throughout our transformations.

Note that after our transformations, the wrist coordinates are constantly equal to 0. For this reason, we did not analyze it's association with the other variables.

#### 5) DIMENSIONALITY REDUCTION

We first standardized our data using Scikit-Learn [13].

Then, we tried out the following dimensionality techniques:

- PCA [14]
- t-SNE [15]
- UMAP [16]
- PCA using a linear kernel, a polynomial kernel, a radial basis function (RBF) kernel, a sigmoid kernel and a cosine kernel. [17]

When using two and three components, we obtained interesting results using t-SNE (with a perplexity set to 30) and UMAP. However, the objective of our project is to be able to use it for live inference. When we integrated these reducers to our process, it slowed it down to severely. Therefore, we decided not to use them.

We obtained poor results when using two and three components and PCA with no kernel, a linear kernel, or a polynomial kernel.

We obtained interesting looking results when using two and three components and PCA with an RBF kernel, a sigmoid kernel, and with a cosine kernel. Among these three, the best was the one using a cosine-base kernel with PCA and 3 components.

Therefore, we will use the Scikit-learn implementation of the Principal Component Analysis (PCA) technique using a cosine-based kernel [17] to reduce the dimensionality of our dataset.

See figures 4 and 5 in the Additional Figures Section for the visualizations using 2 components for each of the reducers.

To choose the number components, we decided to create 14 reducers with the number of components being numbers varying between 1 and 30. Then we transformed our data using each of these reducers. Then we created 128 K Nearest Neighbors models based for each of these 14 datasets. Each of the models had a value of K going from 1 to 128. We then trained each of the models, and evaluated their accuracy. We then chose the number of components that had the model with the best accuracy.

The number of components we chose is 10.

### III. K NEAREST NEIGBORS MODEL

#### A. TRAINING AND HYPERPARAMETER TUNING

We use the Scikit-Learn implementation K Nearest Neighbors (KNN) algorithm [14] to classify our data.

We chose this model because it allowed us to focus on the data preparation transformations and spend less time on tuning hyperparameters. In fact the only hyperparameter we had to choose was k. As mentioned previously, we tried a large number of combinations of k, the number of neighbors, and the number of components used in the dimensionality reducer.

We obtained the best results when using 10 components in the Cosine-base Kernel PCA reducer, with 1 neighbor in the KNN algorithm.

In figure 6 of the Additional Figures section, we can see the accuracy scores obtained for different values of k and 10 components in the reducer.

#### B. EVALUATION AND METRICS

As this a classification problem, we chose the following metrics: Accuracy, Recall, Precision and F1 Score.

Here are the results we obtained for our model:

	Accuracy	Recall	Precision	F1
Score	0.9907	0.9849	0.9970	0.9909

With the model trained in this project, we were able to implement a live-inference tool that classified the “A” and “B” signs using a user’s webcam at a satisfactory rate. It is worth noting that because of Mediapipe’s versatility, this model should be portable to many devices.

### IV. CONCLUSIONS

This project’s results indicate that the transformations we have been developing yield data that is “learnable”. Now in future projects it feels more reasonable to focus on the model and we can explore more complicated models which may need more attention to tune hyperparameters.

The next step is to try to work on more than two classes. First by adding other classes for static signs and later including dynamic signs. When dynamic signs are incorporated, I hope to add other transformations or modify the model to work on more than one frame at once.

If in the future we decide to continue with this project and move past alphanumeric signs into more general-use signs, we could try incorporating the pose recognition and face landmarking Mediapipe models to consider the hand’s relative position to specific body parts.

Another point that needs further exploration is finding another dimensionality reduction technique that is light enough to be used for live inference but that yields better results in terms of intra-class relationships than PCA.

### V. REFERENCES

- [1] N. Mohamed, M. B. Mustafa and N. Jomhari, "A Review of the Hand Gesture Recognition System: Current Progress and Future Directions," *IEEE Access*, 2021.
- [2] G. Garcia-Bautista, F. Trujillo-Romero and S. O. Caballero-Morales, "Mexican Sign Language Recognition Using Kinect and

- Data Time Warping Algorithm," *IEEE Xplore*, 2017.
- [3] M. E. Trejo Rodriguez, Modelos computacionales aplicados a la Lengua de Señas Mexicana, 2019.
  - [4] M. E. Trejo Rodriguez, O. Oubram, B. Ali and N. Lakouari, "Mexican Sign Language's Dactylogy and Ten First Numbers - Labeled images and videos. From person #1 to #5," 30 May 2023. [Online]. Available: <https://data.mendeley.com/datasets/5s4mt7xrd9/1>. [Accessed 13 November 2024].
  - [5] M. E. Trejo Rodriguez, O. Oubram, B. Ali and N. Lakouari, "Mexican Sign Language's Dactylogy and Ten First Numbers - Labeled images and videos. From person #6 to #11," 30 May 2023. [Online]. Available: <https://data.mendeley.com/datasets/67htnzmwbb/1>. [Accessed 13 November 2024].
  - [6] C. d. M. INDEPEDI, "Diccionario de Lengua de Señas Mexicana, Ciudad de México," 25 September 2017. [Online]. Available: [https://pdh.cdmx.gob.mx/storage/app/media/banner/Dic\\_LSM%202.pdf](https://pdh.cdmx.gob.mx/storage/app/media/banner/Dic_LSM%202.pdf). [Accessed 17 November 2024].
  - [7] E. R. d. Molteni, "Vocabulario Animales LSM," [Online]. Available: <https://youtu.be/0XPEfoqRnXo?si=6PpcYAF7SPoFATqV&t=67>. [Accessed 17 November 2024].
  - [8] E. R. d. Molteni, "Relaciones Familiares en LSM," [Online]. Available: <https://youtu.be/NWEJI39YaKA?si=ENqTmVS5zdqN77tX&t=60>. [Accessed 17 November 2024].
  - [9] OpenCV, "OpenCV - Open Computer Vision Library," [Online]. Available: <https://opencv.org>. [Accessed 05 11 2024].
  - [10] Google, "Mediapipe," [Online]. Available: <https://developers.google.com/mediapipe..> [Accessed 9 April 2024].
  - [11] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang and M. Grundmann, "MediaPipe Hands: On-device Real-time Hand Tracking," Google Research, 2020.
  - [12] Apache Spark Foundation, "Apache Spark™ - Unified Engine for large-scale data analytics," [Online]. Available: <https://spark.apache.org>. [Accessed 05 11 2024].
  - [13] Scikit-Learn, "StandardScaler," [Online]. Available: <https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html>. [Accessed November 17 2024].
  - [14] Scikit-Learn, "KNeighborsClassifier -- scikit-learn 1.6.dev0 documentation," [Online]. Available: <https://scikit-learn.org/dev/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>. [Accessed 05 Nov 2024].
  - [15] Scikit-Learn, "TSNE," [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>. [Accessed 17 November 2024].
  - [16] L. McInness, "How to Use UMAP," [Online]. Available: [https://umap-learn.readthedocs.io/en/latest/basic\\_usage.html](https://umap-learn.readthedocs.io/en/latest/basic_usage.html). [Accessed 17 November 2024].
  - [17] Scikit-Learn, "KernelPCA," [Online]. Available: <https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.KernelPCA.html#r396fc7d924b8-1>. [Accessed 17 November 2024].
  - [18] Scikit-Learn, "PCA -- scikit-learn 1.6.dev0 documentation," [Online]. Available: <https://scikit-learn.org/dev/modules/generated/sklearn.decomposition.PCA.html>. [Accessed 5 Nov 2024].



VI. ADDITIONAL FIGURES

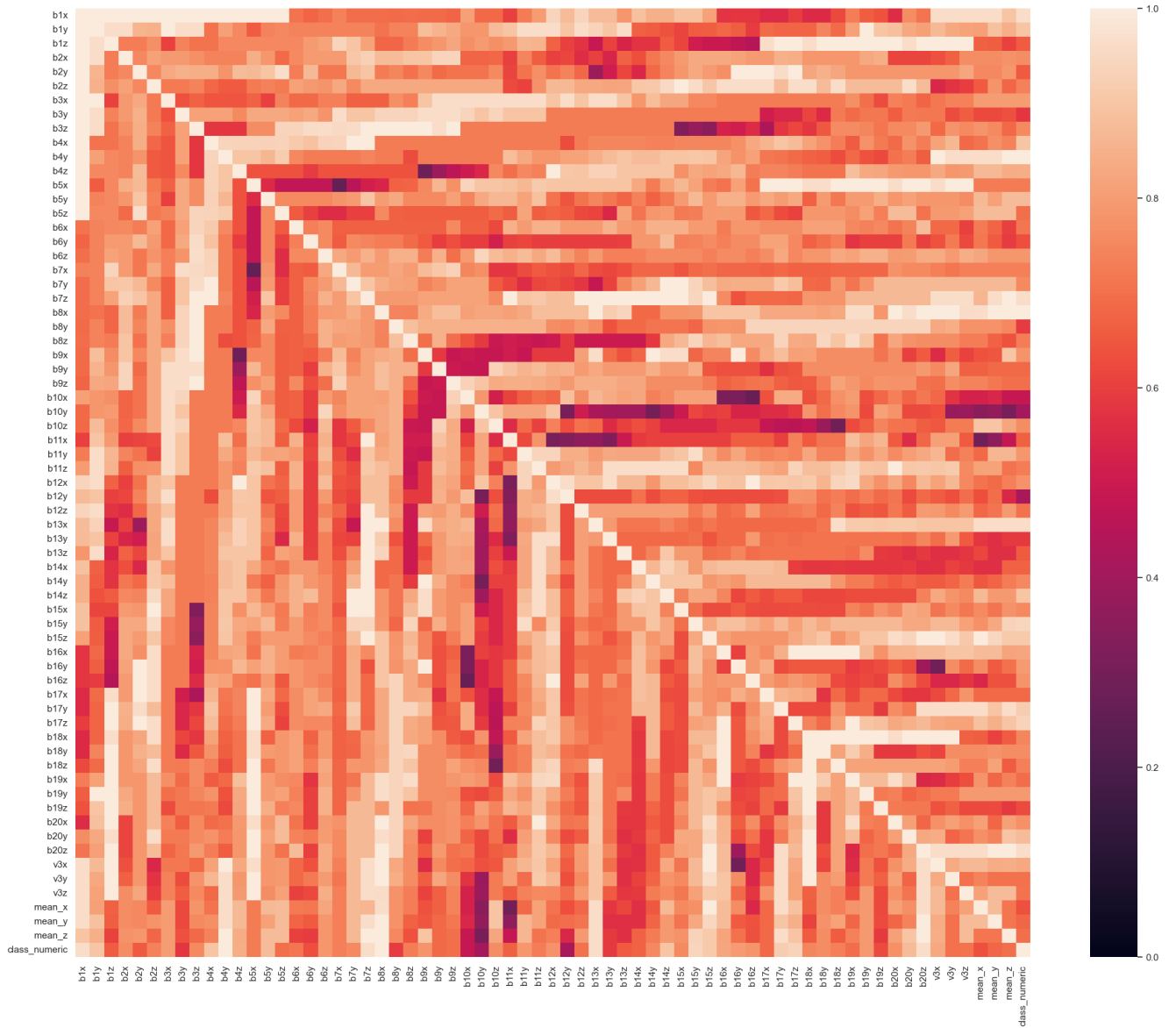


Figure 3: Maximal Information Coefficient score matrix between pairs of every non-constant variable in our transformed dataset.

Comparison of PCA, t-SNE, UMAP 2-component projections of our dataset

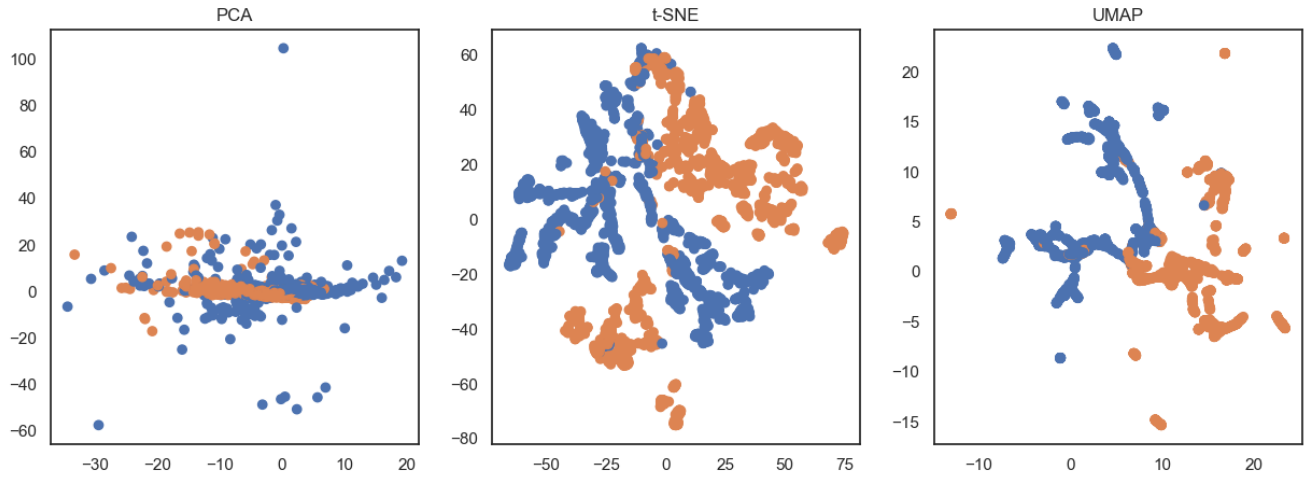


Figure 4

Comparison of linear, polynomial, rbf, sigmoid and cosine-kernel PCA 2-component projections of our dataset

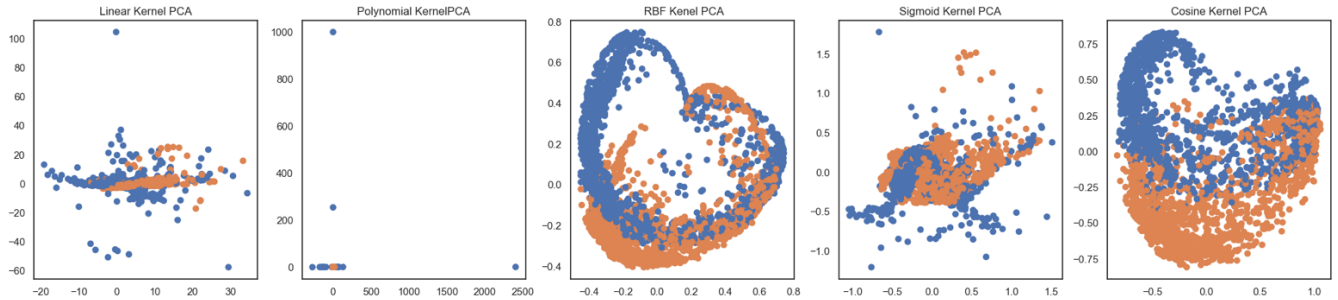


Figure 6

Accuracy score vs k-value for n=10

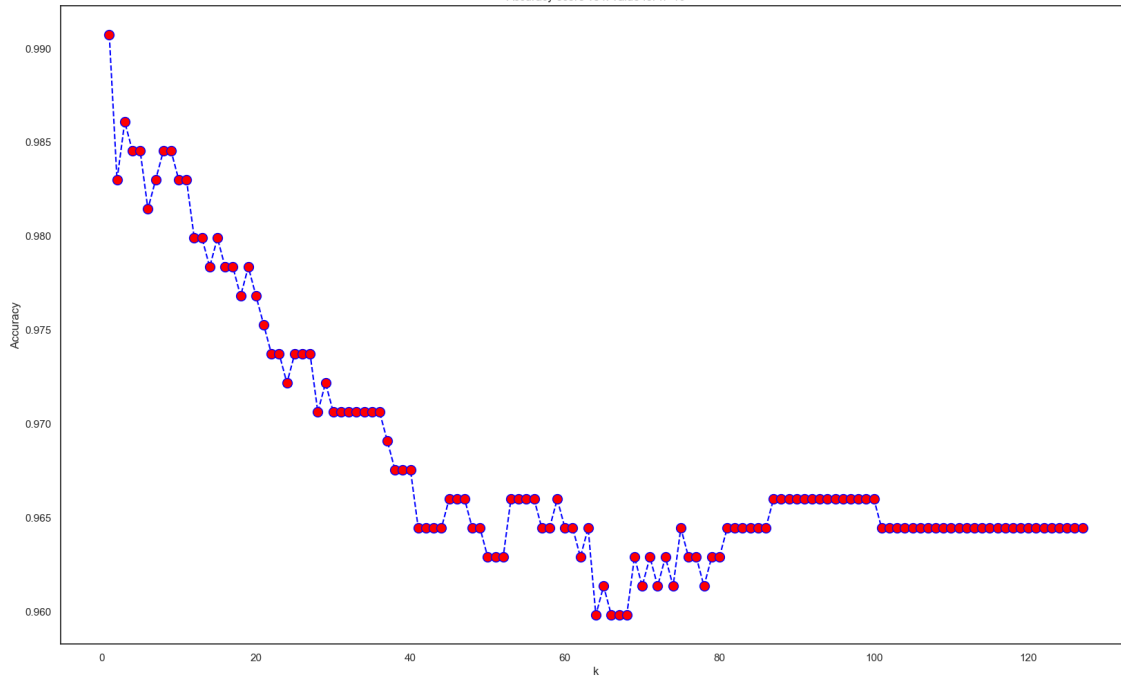


Figure 5